O‘ZBEKISTON RESPUBLIKASI OLIY VA O‘RTA MAXSUS TA’LIM VAZIRLIGI

O‘RTA MAXSUS, KASB-HUNAR TA’LIMI MARKAZI

SH. R. UBAYDULLAYEVA, K. Z. ABIDOV, Z. K. ABIDOVA

HISOBLASH VA MIKROPROTSESSORLI TEXNIKA ASOSLARI

Kasb-hunar kollejlari uchun o‘quv qo‘llanma

Qayta ishlangan va to‘ldirilgan 4-nashri

Toshkent — «ILM ZIYO» — 2016

UO‘K: 007(075)
KBK 32.973.26-04
U13

Oliy va o‘rta maxsus, kasb-hunar ta’limi ilmiy-metodik birlashmalari faoliyatini muvofiqlashtiruvchi Kengash tomonidan nashrga tavsiya etilgan.

Taqrizchilar: **F.A. Qurbonov** — texnika fanlari nomzodi, dotsent, Buxoro politexnika kasb-hunar kolleji o‘qituvchisi;
O.I. Jalolov — Buxoro davlat universiteti «Amaliy matematika va axborot texnologiyalari» kafedrasi dotsenti;
P.R. Bozorov — Buxoro muhandislik-texnologiya instituti «Texnologik jarayonlarni boshqarishning axborot-kommunikatsiya tizimlari» kafedrasi dotsenti.

Mazkur o‘quv qo‘llanma «Hisoblash va mikroprotsessorli texnika asoslari» fanidan namunaviy dastur asosida yozilgan bo‘lib, unda zamonaviy hisoblash va mikroprotsessorli texnikaning arifmetik va mantiqiy asoslari bayon etilgan.

O‘quv qo‘llanma kasb-hunar kollejlari o‘quvchilari uchun mo‘ljallangan.

ISBN 978-9943-16-355-3

SO‘ZBOSHI

Axborotlarni qayta ishlash sohasida insoniyat to‘rt axborot inqilobini boshidan kechirdi. Bu axborot inqiloblari jamiyatdagi ijtimoiy-iqtisodiy munosabatlarni har safar yangi bosqichga ko‘tarishga xizmat qildi.

So‘nggi axborot inqilobi (XX asrning 70-yillari) mikroprotsessorli texnologiyaning ixtiro qilinishi va shaxsiy kompyuterlarning yaratilishi bilan bevosita bog‘liqdir. Unga ko‘ra hozirga qadar yanada takomillashtirilgan mikroprotsessorlar va integral sxemalarda ishlovchi zamonaviy kompyuterlar, kompyuter tarmoqlari va axborot kommunikatsiyalari ishlab chiqarilmoqda.

«Hisoblash va mikroprotsessorli texnika asoslari» fanini o‘rganishning muhimligi talabalarda zamonaviy mikroprotsessorlarning arifmetik va mantiqiy asoslarini hamda ularning funksional-strukturaviy tashkil etilishining nazariy va amaliy asoslari haqidagi bilimlarni to‘la shakllantirish zarurligi bilan izohlanadi.

O‘quv qo‘llanma ayni shu dolzarb vazifalarni hal etishga xizmat qiladi, degan umiddamiz. U bir-biridan mustaqil va, ayni paytda, mantiqiy izchil bog‘liq bo‘lgan ikki qismdan iborat.

O‘quv qo‘llanmaning birinchi bobi hisoblash va mikroprotsessorli texnikaning arifmetik asoslarini o‘rganishga bag‘ishlangan bo‘lsa, uning ikkinchi bobida hisoblash va mikroprotsessorli texnikaning mantiqiy asoslari bayon etilgan. Uchinchi bobda kombinatsiyalashgan qurilmalarning sxemotexnikasi xususida ma’lumotlar keltirilgan. O‘quv qo‘llanmaning to‘rtinchi bobi ketma-ket raqamli qurilmalarning sxemotexnikasiga bag‘ishlangan. Beshinchi bobda esa mikroprotsessorning arxitekturasi bayon etilgan.

O‘quv qo‘llanmani yaratishda «Hisoblash va mikroprotsessorli texnikaning asoslari» o‘quv fani uchun ishlab chiqilgan namunaviy dastur asos qilib olindi.

Mualliflar

1- bob. HISOBLASH VA MIKROPROTSESSORLI TEXNIKANING ARIFMETIK ASOSLARI

1.1. POZITSION SANOQ SISTEMALARI

Ma'lumki, EHM axborotlar ustida ish olib borish uchun qo'llaniladi. Axborotlarni kodlash, qabul qilish, ularni qayta ishlash va uzatish jarayonlari EHMda ma'lum qoidalar asosida amalga oshiriladi. Bu qoidalarni to'la tushunib olganimizdagina, biz EHMlar dunyosi qanday «quriladi» degan savolga batafsil javob berishimiz mumkin. Buning uchun, avvalo, turli sanoq sistemalari bilan tanishib chiqishimiz lozim.

O'nlik sanoq sistemasi o'nta raqam: 0, 1, 2, ..., 9 dan iborat, uning asosi (ya'ni raqamlar soni) $P = 10$. Bu sanoq sistemasidan biz kundalik turmushimizda keng foydalanamiz.

Sakkizlik sanoq sistemasi sakkizta raqam: 0, 1, 2, ..., 7 dan iborat, uning asosi $P = 8$.

Ikkilik sanoq sistemasi ikkita raqam: 0, 1 dan iborat, uning asosi $P = 2$.

O'n oltilik sanoq sistemasi quyidagi o'n oltita belgidan iborat: 0, 1, 2, ..., 9, A, B, C, D, E, F, asosi $P = 16$. Bunda A = 10, B = 11, C = 12, D = 13, E = 14, F = 15 ni bildiradi.

Umumiy holda, pozitsion sistemadagi sonlarni quyidagi ko'rinishda yozish mumkin:

$$A = a_n a_{n-1} \ldots a_2 a_1 a_0, \quad a_{-1} a_{-2} \ldots a_{-(m-1)} a_{-m}. \tag{1.1}$$

Bu ko'rinishdagi ixtiyoriy sonni quyidagi formula yordamida yoyib yozish mumkin:

$$\begin{aligned} A = a_n a_{n-1} \ldots a_2 a_1 a_0, \quad a_{-1} a_{-2} \ldots a_{-(m-1)} a_{-m} = \\ = a_n \cdot P^n + a_{n-1} \cdot P^{n-1} + \ldots + a_1 \cdot P^1 + \\ + a_0 \cdot P^0 + a_{-1} \cdot P^{-1} + a_{-2} \cdot P^{-2} + \ldots + a_{-m} \cdot P^{-m}, \end{aligned} \tag{1.2}$$

bu yerda: $0 < a_k < P$ – raqamlar; P – sanoq sistemasining asosi, $m < k < n$ – razryad (xona) raqami.

1-qoida. O'nlik sanoq sistemasidagi butun sonni ikkilik (sakkizlik yoki o'n oltilik) sanoq sistemasiga o'tkazish uchun shu sonni va keyingi hosil bo'lgan bo'linmalarni o'tilayotgan sanoq sistemasining asosiga, ya'ni 2 ga (8 ga yoki 16 ga) toki oxirgi bo'linma 2 dan (8 dan yoki 16 dan) kichik bo'lguncha, ketma-ket bo'lish lozim. So'ngra oxirgi natijani va hosil bo'lgan qoldiqlarni teskari tartibda yozib chiqish kerak.

2-qoida. O‘nlik sanoq sistemasidagi kasr sonni ikkilik (sakkizlik yoki o‘n oltilik) sanoq sistemasiga o‘tkazish uchun shu sonni va keyingi hosil bo‘lgan kasr qismlarni o‘tilayotgan sanoq sistemasining asosiga, ya’ni 2 ga (8 ga yoki 16 ga) ketma-ket ko‘paytirish lozim. Natijada hosil bo‘lgan butun qismlarni to‘g‘ri tartibda yozib chiqish kerak. Ko‘paytirish jarayonini 2 holda to‘xtatish mumkin:

1) kasr qismida nollar hosil bo‘lgan taqdirda;

2) berilgan razryad aniqligiga erishilgan holda.

3-qoida. O‘nlik sanoq sistemasidagi aralash sonni ikkilik (sakkizlik yoki o‘n oltilik) sanoq sistemasiga o‘tkazish uchun shu sonning butun qismini 1-qoidaga muvofiq alohida, kasr qismini 2-qoidaga muvofiq alohida o‘tkazib, hosil bo‘lgan natijalarni qo‘shish lozim.

4-qoida. Ikkilik, sakkizlik yoki o‘n oltilik sanoq sistemasidagi sonni o‘nlik sanoq sistemasiga o‘tkazish uchun (1.2) formuladan foydalanish mumkin.

5-qoida. Sonlarni sakkizlik sanoq sistemasidan ikkilik sanoq sistemasiga o‘tkazish uchun sakkizlik sonning har bir raqamini uch xonali ikkilik son yoki *triada* bilan almashtirish kerak (1-jadval).

Sonlarni ikkilik sanoq sistemasidan sakkizlik sanoq sistemasiga o‘tkazish uchun sonni verguldan chap va o‘ng tomoniga qarab 3 xonadan (triadalardan) iborat guruhlarga ajratish va har bir triadani sakkizlik sanoq sistemasiga mos raqamlar bilan yozish talab etiladi, bunda chetdagi to‘liqmas triadalar nollar bilan to‘ldiriladi.

6-qoida. Sonni o‘n oltilik sanoq sistemasidan ikkilik sanoq sistemasiga o‘tkazish uchun *tetradalardan* foydalanish mumkin (2-jadval). O‘n oltilik sonning har bir raqamini to‘rt xonali ikkilik son yoki tetrada bilan almashtirish kerak.

Sonni ikkilik sanoq sistemasidan o‘n oltilik sanoq sistemasiga o‘tkazish uchun, sonni verguldan chapdan o‘ng tomoniga qarab 4 xonadan (tetradalardan) iborat guruhlarga ajratish va har bir tetradani o‘n oltilik sanoq sistemasida ifodalash mumkin. Bu holda ham chetdagi to‘lmagan tetradalar nollar bilan to‘ldiriladi.

1-jadval

Sakkizlik sonlar	0	1	2	3	4	5	6	7
Ikkilik sonlar	0	1	10	11	100	101	110	111
Triadalar	000	001	010	011	100	101	110	111

2-jadval

O‘n oltilik sonlar	Ikkilik sonlar	Tetradalar
1	1	0001
2	10	0010
3	11	0011
4	100	0100
5	101	0101
6	110	0110
7	111	0111
8	1000	1000
9	1001	1001
A	1010	1010
B	1011	1011
C	1100	1100
D	1101	1101
E	1110	1110
F	1111	1111

Mazkur qoidalar asosida butun, kasr va aralash sonlarni bir sanoq sistemasidan boshqasiga o‘tkazish bo‘yicha misollarni ko‘rib chiqamiz.

1.1-misol. 25_{10} sonini ikkilik va sakkizlik sanoq sistemalariga o‘tkazing.

Yechish.

a)
$$\begin{array}{r|l} 25 & 2 \\ \hline 24 & 12 \\ \hline 1 & 12 \\ \hline 0 & 6 \\ \hline 0 & 2 \\ \hline 1 & \end{array}\qquad\begin{array}{r|l} 12 & 2 \\ \hline 6 & 2 \\ \hline 3 & 2 \\ \hline 2 & 1 \\ \hline & \end{array}$$

o‘qilish yo‘nalishi — oxirgi bo‘linma

b)
$$\begin{array}{r|l} 25 & 8 \\ \hline 24 & 3 \\ \hline 1 & \end{array}$$

o‘qilish yo‘nalishi

Demak, $(25)_{10} = (11001)_2 = (31)_8$.

Hosil bo‘lgan natijalarni qaytib o‘tish yo‘li orqali tekshirish mumkin. Buning uchun (1.2) ifodadan foydalanib, ikkilik va sakkizlik sanoq sistemalarida hosil bo‘lgan sonlarni o‘nlik sanoq sistemasiga mos ravishda qayta o‘tkazamiz.

a) $(11001)_2 = 1 \cdot 2^4 + 1 \cdot 2^3 + 0 \cdot 2^2 + 0 \cdot 2^1 + 1 \cdot 2^0 =$
$= 16 + 8 + 0 + 0 + 1 = (25)_{10}$.

b) $(31)_8 = 3 \cdot 8^1 + 1 \cdot 8^0 = (25)_{10}$.

1.2-misol. 2^{-4} aniqlikda berilgan 0,3125 o‘nlik kasr sonni ikkilik sanoq sistemasiga o‘tkazing.

Yechish.

× 0,3125	× 0,6250	× 0,2500	× 0,5000
2	2	2	2
0,6250	1,2504	0,5000	1,0000

o‘qilish yo‘nalishi

Demak, $(0,3125)_{10} = (0,0101)_2$.

(1.2) ifodadan foydalangan holda, chiqqan sonni o‘nlik sanoq sistemasiga qayta o‘tkazib, natijani tekshiramiz:

$$(0,0101)_2 = 0 \cdot 2^{-1} + 1 \cdot 2^{-2} + 0 \cdot 2^{-3} + 1 \cdot 2^{-4} =$$
$$= 1/4 + 1/16 = 5/16 = (0,3125)_{10}.$$

1.3-misol. 2^{-3} aniqlikda berilgan $159,75_{10}$ aralash sonni ikkilik sanoq sistemasiga o‘tkazing.

Yechish. Yuqorida keltirilgan qoidalarga tayangan holda, o‘nlik aralash sonning butun qismini alohida va kasr qismini alohida ikkilik sanoq sistemasiga o‘tkazib, natijalarni qo‘shamiz:

$$(159)_{10} = (10011111)_3; \quad (0,75)_{10} = (0,11)_2;$$

demak, $(159,75)_{10} = (10011111,11)_2$.

1.4-misol. Sakkizlik sanoq sistemasida berilgan $325,27_8$ sonni ikkilik sanoq sistemasiga o‘tkazing.

Yechish. 1-jadvaldan foydalanib, aralash sonni triadalar bo‘yicha yozib chiqamiz va quyidagini hosil qilamiz:

$$325,27_8 = (011\ 010\ 101,010\ 111)_2 = 11010101,010111_2.$$

1.5-misol. Berilgan (1011101,01101) ikkilik sonni sakkizlik sanoq sistemasiga o‘tkazing.

Yechish.

$$(1011101,01101)_2 = 001\ 011\ 101,011\ 010 = (135,32)_8.$$

1.6-misol. O‘n oltilik sanoq sistemasidagi $(C876,F3)_{16}$ sonni ikkilik sanoq sistemasiga o‘tkazing.

Yechish. 2-jadvaldan foydalanib, quyidagi natijaga erishamiz:

$$(C876,F3)_{16} = (1100\ 1000\ 0111\ 0110,1111\ 0011)_{16-2} =$$
$$= (1100100001110110,11110011)_2.$$

1.7-misol. $(1011101101,101101101)_2$ ikkilik sonni o‘n oltilik sanoq sistemasiga o‘tkazing.

Yechish. 6-qoidaga binoan aralash ikkilik sonni tetradalarga bo‘lib ifodalash orqali quyidagi natijani hosil qilish mumkin:

$$(1011101101,\ 101101101)_2 =$$
$$= (0010\ 1110\ 1101,1011\ 0110\ 1000)_{16-2} = (2ED,B68)_{16}.$$

Nazorat savollari va topshiriqlari

1. Ixtiyoriy sanoq sistemasida berilgan butun son o‘nlik sanoq sistemasida qanday yoziladi?
2. Ixtiyoriy sanoq sistemasida berilgan kasr son o‘nlik sanoq sistemasida qanday yoziladi?
3. Ixtiyoriy P asosli sanoq sistemasidan Q asosli sanoq sistemasiga o‘tish qoidasini ayting.
4. D^{-4} (D – sanoq sistemasining asosi) aniqlik bilan sonlarni bir sanoq sistemasidan boshqasiga o‘tkazing:
   a) $(287,3504)_{10} = (?)_8 = (?)_{16} = (?)_2$;
   b) $(39,4571)_{10} = (?)_{16} = (?)_2 = (?)_8$.

1.2. TURLI POZITSION SANOQ SISTEMALARIDA ARIFMETIK AMALLARNI BAJARISH

Ikkilik sanoq sistemasida arifmetik amallarni bajarish qoidalari. Ma’lumki, ikkilik sanoq sistemasida faqat ikkita raqam: 0 va 1 qatnashadi. Ikkilik sanoq sistemasida qo‘shish, ayirish va ko‘paytirish amallari quyidagi qoidalar asosida amalga oshiriladi:

a)	b)	d)
$0 + 0 = 0$	$0 - 0 = 0$	$0 \cdot 0 = 0$
$0 + 1 = 1$	$1 - 0 = 1$	$0 \cdot 1 = 0$
$1 + 0 = 1$	$1 - 1 = 0$	$1 \cdot 0 = 0$
$1 + 1 = 10$	$10 - 1 = 1$	$1 \cdot 1 = 1$

Endi ikkilik sanoq sistemasida turli arifmetik amallarni bajarishga doir misollarni ko‘rib chiqamiz.

1.8-misol. 1010_2 va 1011_2 sonlarining yig‘indisini toping.

Yechish. Sonlarni bir ustunga yozib, umumiy qoida bo‘yicha qo‘shamiz:

$$\begin{array}{r} 1010_2 \\ +\ 1011_2 \\ \hline 10101_2 \end{array}$$

Javob: 10101_2.

1.9-misol. $1011,01_2$ va $101,101_2$ sonlarining yig‘indisini toping.

Yechish.
$$\begin{array}{r} 1011,01_2 \\ +\ 101,101_2 \\ \hline 10000,111_2 \end{array}$$
Javob: $10000,111_2$.

1.10-misol. $101,01_2$ va $10,10_2$ sonlarining ayirmasini toping.

Yechish.
$$\begin{array}{r} 101,01_2 \\ -\ 10,10_2 \\ \hline 10,11_2 \end{array}$$
Javob: $10,11_2$.

1.11-misol. $1011,11_2$ va $101,101_2$ sonlarining ayirmasini toping.

Yechish.
$$\begin{array}{r} 1011,11_2 \\ -\ 101,101_2 \\ \hline 110,001_2 \end{array}$$
Javob: $110,001_2$.

1.12-misol. 10111_2 va 101_2 sonlari ko‘paytmasini toping.

Yechish. Ikkilik sanoq sistemasida sonlarni ko‘paytirish o‘nlik sanoq sistemasida qabul qilingan qoidalar asosida bajariladi:
$$\begin{array}{r} \times\ 10111_2 \\ 101_2 \\ \hline 10111 \\ +\ 00000 \\ \hline 10111 \\ \hline 1110011_2 \end{array}$$
Javob: 1110011_2.

1.13-misol. $10,11_2$ va $111,11_2$ sonlarining ko‘paytmasini toping.

Yechish.
$$\begin{array}{r} \times\ 10,11_2 \\ 111,11_2 \\ \hline 1011 \\ \hline 1011 \\ +\ 1011 \\ 1011 \\ 1011 \\ \hline 10101,0101_2 \end{array}$$
Javob: $10101,0101_2$.

Sakkizlik sanoq sistemasida arifmetik amallarni bajarish. Ma’lumki, sakkizlik sanoq sistemasida sonlarni yozish uchun hammasi bo‘lib sakkizta raqam: 0, 1, 2, 3, 4, 5, 6, 7 dan foydalaniladi, chunki uning asosi $q = 8$ dir. Sakkizlik sanoq sistemasida qo‘shish, ayirish va ko‘paytirish amallarini quyida keltirilgan jadvallar ma’lumotlari asosida bajarish mumkin. Sakkizlik sanoq sistemasida

qo‘shish amalini bajarish uchun 3-jadvaldan foydalanish mumkin.

Sakkizlik sanoq sistemasida ko‘paytirish amalini bajarish uchun 4-jadval ma’lumotlari xizmat qiladi.

Bu jadvallar asosida sakkizlik sanoq sistemasida arifmetik amallarni bajaramiz.

1.14-misol. 732_8 va 324_8 sonlarining yig‘indisi va ayirmasini toping.

Yechish. Qo‘shish va ayirish amallari, odatdagidek, sonlarni bir ustunga yozish orqali bajariladi:

a) $+\begin{array}{r} 732_8 \\ 324_8 \\ \hline 1256_8 \end{array}$ b) $-\begin{array}{r} 732_8 \\ 324_8 \\ \hline 406_8 \end{array}$

Javob: 1256_8 va 406_8.

1.15-misol. $363{,}32_8$ va $23{,}7_8$ sonlarining yig‘indisi va ayirmasini toping.

Yechish. a) $+\begin{array}{r} 363{,}32_8 \\ 23{,}7_8 \\ \hline 407{,}22_8 \end{array}$ b) $-\begin{array}{r} 363{,}32_8 \\ 23{,}7_8 \\ \hline 337{,}42_8 \end{array}$

Javob: $407{,}22_8$ va $337{,}42_8$.

1.16-misol. Sakkizlik sanoq sistemasida 23_8 sonini 148_8 soniga ko‘paytiring.

Yechish. Sonlarni sakkizlik sanoq sistemasida ko‘paytirish uchun ularni, odatdagidek, bir ustunga yozib, ketma-ket ko‘paytiramiz va hosil bo‘lgan ko‘paytmalarni qo‘shib chiqamiz:

$$\begin{array}{r} \times\ 23_8 \\ 14_8 \\ +\ 114 \\ 23 \\ \hline 344_8 \end{array}$$

Javob: 344_8.

1.17-misol. Sakkizlik sanoq sistemasidagi $23{,}4_8$ va $12{,}2_8$ sonlarini o‘zaro ko‘paytiring.

Yechish.

$$\begin{array}{r} \times\ 23{,}4_8 \\ 12{,}2_8 \\ 470 \\ +\ 470 \\ 234 \\ \hline 307{,}70_8 \end{array}$$

Javob: $307{,}70_8$.

3-jadval

+	0	1	2	3	4	5	6	7
0	0	1	2	3	4	5	6	7
1	1	2	3	4	5	6	7	10
2	2	3	4	5	6	7	10	11
3	3	4	5	6	7	10	11	12
4	4	5	6	7	10	11	12	13
5	5	6	7	10	11	12	13	14
6	6	7	10	11	12	13	14	15
7	7	10	11	12	13	14	15	16

4-jadval

×	0	1	2	3	4	5	6	7
0	0	0	0	0	0	0	0	0
1	0	1	2	3	4	5	6	7
2	0	2	4	6	10	12	14	16
3	0	3	6	11	14	17	22	25
4	0	4	10	14	20	24	30	34
5	0	5	12	17	24	31	36	43
6	0	6	14	22	30	36	44	52
7	0	7	16	25	34	43	52	61

O‘n oltilik sanoq sistemasida arifmetik amallarni bajarish. Sonlarni o‘n oltilik sanoq sistemasida ifodalash uchun o‘n oltita belgi: 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F dan foydalaniladi. O‘n oltilik sanoq sistemasida o‘n olti soni yaxlit bir butunlikni tashkil etgani uchun 10 ga tengdir.

O‘n oltilik sanoq sistemasida arifmetik amallarni bajarish tartibi o‘nlik sanoq sistemasidagi kabi amalga oshiriladi. Agar o‘n oltilik sanoq sistemasidagi qo‘shiluvchilar ko‘p xonali bo‘lsa, unda 5-jadval ma’lumotlaridan foydalanish mumkin.

Jadval ma’lumotlari asosida o‘n oltilik sanoq sistemasida qo‘shish va ayirish amallariga misollar keltiramiz.

1.18-misol. $A3B9,EF_{16}$ soniga $342,A1_{16}$ sonini qo‘shing.

Yechish.
$$\begin{array}{r} A3B9,EF_{16} \\ +\ 342,A1_{16} \\ \hline A6FC,90_{16} \end{array}$$

Javob: $A6FC,90_{16}$.

5-jadval

O‘n oltilik sanoq sistemasida qo‘shish jadvali

+	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
1	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	10
2	2	3	4	5	6	7	8	9	A	B	C	D	E	F	10	11
3	3	4	5	6	7	8	9	A	B	C	D	E	F	10	11	12
4	4	5	6	7	8	9	A	B	C	D	E	F	10	11	12	13
5	5	6	7	8	9	A	B	C	D	E	F	10	11	12	13	14
6	6	7	8	9	A	B	C	D	E	F	10	11	12	13	14	15
7	7	8	9	A	B	C	D	E	F	10	11	12	13	14	15	16
8	8	9	A	B	C	D	E	F	10	11	12	13	14	15	16	17
9	9	A	B	C	D	E	F	10	11	12	13	14	15	16	17	18
A	A	B	C	D	E	F	10	11	12	13	14	15	16	17	18	19
B	B	C	D	E	F	10	11	12	13	14	15	16	17	18	19	20
C	C	D	E	F	10	11	12	13	14	15	16	17	18	19	20	21
D	D	E	F	10	11	12	13	14	15	16	17	18	19	20	21	22
E	E	F	10	11	12	13	14	15	16	17	18	19	20	21	22	23
F	F	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24

1.19-misol. $12FE,AB_{16}$ soniga $ABC,13_{16}$ sonini qo‘shing.

Yechish.
$$\begin{array}{r} 12FE,AB_{16} \\ +\ ABC,13_{16} \\ \hline 1DBA,BE_{16} \end{array}$$

Javob: $1DBA, BE_{16}$.

1.20-misol. $12FE,AB_{16}$ sonidan $ABC,13_{16}$ sonini ayiring.

Yechish.
$$\begin{array}{r} 12FE,AB_{16} \\ -\ ABC,13_{16} \\ \hline 842,98_{16} \end{array}$$

Javob: $842,98_{16}$.

6-jadval ma’lumotlari asosida o‘n oltilik sanoq sistemasida ko‘paytirish amalini bajaramiz.

6-jadval

O‘n oltilik sanoq sistemasida ko‘paytirish jadvali

×	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	1	2	3	4	5	6	7	8	9	A	B	c	D	E	10
2	2	4	6	8	A	C	E	10	12	14	16	18	1A	1C	1E
3	3	6	9	C	F	12	15	18	1B	1E	21	24	27	2A	2D
4	4	8	C	10	14	18	1C	20	24	28	2C	30	34	38	3C
5	5	A	F	14	19	1E	23	28	2D	32	37	2C	41	46	4B
6	6	C	12	18	1E	24	2A	30	36	3C	42	48	4E	54	5A
7	7	E	15	1C	23	2A	31	38	3E	46	40	54	58	62	69
8	8	10	18	20	28	30	38	40	48	50	58	60	68	70	78
9	9	12	1B	24	2D	36	3F	48	51	5A	63	6C	75	7E	87
A	A	14	1E	28	32	3C	46	50	5A	64	6E	78	82	8C	96
B	B	16	21	2C	37	42	4D	58	63	6E	79	84	8F	9A	A5
C	C	18	24	30	3C	48	54	60	6C	78	84	90	9C	AB	84
D	D	1A	27	34	41	4E	5B	68	75	82	8F	9C	A9	B6	C3
E	E	1C	2A	38	46	54	62	70	7E	8C	9A	A8	B6	C4	D2
F	F	1E	2D	3C	4B	5A	59	78	87	96	A5	B4	C3	D2	El

1.21-misol. $A3E,12_{16}$ sonini $39,A1_{16}$ soniga ko‘paytiring.

Yechish.

$$\begin{array}{r} \times\ A3E,12_{16} \\ \underline{39,A1_{16}} \\ A3E12 \\ +\ 666CB4 \\ 5C2EA2 \\ \underline{1EBA36} \\ 24E43,0B52_{16} \end{array}$$

Javob: $24E43,0B52_{16}$.

1.22-misol. $3A5,1F_{16}$ sonini $14,2_{16}$ soniga ko‘paytiring.

Yechish.

$$\begin{array}{r} \times\ 3A5,1F_{16} \\ \underline{14,2_{16}} \\ 74A3E \\ +\ E947C \\ 3A51F \\ \underline{495B,0FE_{16}} \end{array}$$

Javob: $495B,0FE_{16}$.

Nazorat savollari va topshiriqlari

1. Ikkilik sanoq sistemasida qo‘shish amali qanday bajariladi?
2. Ikkilik sanoq sistemasida ko‘paytirish jadvalini yoddan ayting.
3. Sakkizlik sanoq sistemasida qo‘shish jadvalini yoddan ayta olasizmi?
4. Sakkizlik sanoq sistemasida ko‘paytirish jadvalini yoddan ayting.
5. O‘n oltilik sanoq sistemasida qo‘shish jadvalidan bir necha misollar keltiring.
6. O‘n oltilik sanoq sistemasida ko‘paytirish jadvalidan yoddan misollar keltiring.
7. Uchlik sanoq sistemasi uchun qo‘shish va ko‘paytirish jadvalini tuzing.

1.3. MASHINALARDA SONLARNI TAVSIFLASH NORMALARI. QO‘ZG‘ALUVCHI VA QO‘ZG‘ALMAS VERGULLI SONLAR

EHMlarda ikkilik sonlarning ikki xil ko‘rinishi ishlatiladi. Bular sonlarning normal va tabiiy ko‘rinishlaridir. Sonlarning tabiiy ko‘rinishidan biz kundalik turmushda keng foydalanamiz. Sonlarning tabiiy ko‘rinishida vergulning o‘rni bir joyda, ya’ni uning butun va kasr qismlari orasida joylashganligi sababli bu sonlar *qo‘zg‘almas vergulli sonlar* deyiladi. Qo‘zg‘almas vergulli ko‘rinishda tasvirlangan sonlar ustida amallar juda sodda bajariladi, chunki vergulning o‘rni o‘zgarmaydi. Shuning uchun xonalardagi raqamlarni mos ravishda qo‘shib qo‘yish talab etiladi. Bu usulning kamchiligi ishlatiladigan sonlarning chegaralanganidir, bu esa hisob ishlarini olib borishda ancha qiyinchilik tug‘diradi.

Sonlarni qo‘zg‘aluvchi vergulli ko‘rinishda tasvirlash. Q asosli sanoq sistemasidagi ixtiyoriy a ($a \neq 0$) son *normal forma* deb ataluvchi qo‘zg‘aluvchi vergulli ko‘rinishda bu usulda quyidagicha tasvirlanadi:

$$a = \pm M \cdot Q^{\pm p}, \qquad (1.3)$$

bu yerda: M – shu a sonning *mantissasi* deyiladi, u musbat to‘g‘ri kasrdan iborat, p esa a sonning *tartibi* deyiladi, u butun son bo‘ladi. Q esa sanoq sistemasining asosidir.

Sonlarni bu usulda tasvirlashning kamchiliklari ham mavjud. Unda belgilarning ko‘payib ketishi va shu bilan mos holda arifmetik amallarni bajarish jarayonining murakkablashishi kuzatiladi.

Normal formani aniqlashda mantissaning kattaligiga hech qanday shart qo‘yilmaydi, faqat kasr to‘g‘ri bo‘lsa, yetarli hisoblanadi. Shu sababli tartibning o‘zgarishi bilan mantissaning ham, vergulning ham o‘rni o‘zgarib boradi, bunda vergul go‘yo «suzib» yurgan-

dek tuyuladi. Shuning uchun ko‘pincha normal formadagi sonlar *qo‘zg‘aluvchi vergulli sonlar* deyiladi.

Normal qo‘zg‘aluvchi vergulli sonlar bilan ishlaydigan mashinalarning ikkita kamchiligini ta’kidlash mumkin:

1) sonlarning yozilishi yagona bo‘lmaganligi tufayli arifmetik qurilmani murakkablashtirish zarurati tug‘iladi. Normal sonlarning (formaning) ko‘plab ko‘rinishlaridan biri — mantissaning yuqori razryadi (verguldan keyingi birinchi son) nolga teng bo‘lmagan sonli ko‘rinishi *normallashtirilgan son* deyiladi. Demak, sonlarning boshqa ko‘rinishlari *normallashtirilmagan son* deyiladi.

Mashina xotirasida razryad to‘rining cheklanganligi tufayli mantissaning kichik razryadlari yo‘qolmasligi, ya’ni sonlar tavsifining aniqligi pasaymasligi uchun ular normallashtirilgan ko‘rinishda saqlanadi. Lekin arifmetik amallarni bajarish jarayonida natija normallashmagan bo‘lishi mumkin, shuning uchun mashinalarda sonlarni avtomatik normallashtiruvchi maxsus sxema o‘rnatiladi.

Normallashtirish sonning mantissasi verguldan keyin nechta nolga ega bo‘lsa, shuncha xona chapga surilishi va tartib sonini shuncha birlikka kamaytirishga asoslangan. Normallashtirish operatsiyasini misollarda ko‘ramiz.

1.23-misol. Sonlarni normallashtiring:

$X_1 = 10^5 \cdot 0{,}001183754_{(10)}$ va $X_2 = 10^{1000} \cdot 0{,}0111011011_{(2)}$.

Javob: $X_{1norm} = 10^3 \cdot 0{,}118375 = (3;\ 0{,}118375_{(10)})$;

$X_{2norm} = 10^{111} \cdot 0{,}111011011 = (111;\ 0{,}111011011_{(2)})$;

2) sonlarni normallashtirishning kamchiligi shundaki, unda sonning turli qismlari (mantissa va tartibi) bilan turli amallarni bajarish jarayonida AMQ (arifmetik mantiqiy qurilma) sxemasini murakkablashtirish zarurati tug‘iladi, chunonchi, ikkita sonni ko‘paytirishda ularning mantissalarini ko‘paytirish, tartiblarini esa algebraik qo‘shish kerak bo‘ladi.

Qo‘zg‘almas vergulli sonlar bilan ishlovchi mashinalar ko‘rsatilgan kamchiliklarga ega emas, lekin bu fikr bunday mashinalar faqat ijobiy xarakteristikalarga ega degani emas. Bu mashinalar ham ma’lum darajada kamchiliklarga ega, chunki ular razryad to‘rining chegaralangan o‘lchamida operatsiyalar olib borish uchun kichik diapazondagi sonlarni qabul qiladi.

Qo‘zg‘aluvchi vergulli sonlar rejimida sonlar diapazoni juda keng bo‘lib, u -10^{99} dan $+10^{99}$ gacha oraliqni tashkil etadi. Qo‘zg‘almas vergulli sonlar rejimida esa mashina sonlari diapazondan tashqarida bo‘lish xavfi ancha yuqoridir, chunki barcha mashina

sonlari juda tor oraliqda (-10^3 dan $+10^3$ gacha) berilgan. Shuning uchun qo‘zg‘almas vergulli sonlar bilan ishlovchi mashinalarda razryad to‘ri to‘lib ketishi mumkin, bunda sonning yuqori razryadlari yo‘qoladi, bu esa uning qo‘pol va noaniq akslanishiga olib keladi.

EHMlarda turli amallarni bajarish uchun sonlarni maxsus mashina kodlari bilan kodlashtirish zarur. Sonlar ustida ayirish amalini qo‘shish amaliga almashtirish ehtiyoji paydo bo‘ladi, bunda qo‘shimcha va teskari kodlardan foydalaniladi.

To‘g‘ri kod. Manfiy sonning *to‘g‘ri kodi* deb, uni tabiiy ko‘rinishda tasvirlashga aytiladi va ishorali razryadiga 1 soni qo‘yiladi.

Musbat sonning to‘g‘ri kodi uning tabiiy ko‘rinishiga mos tushadi. $A = 0,a_1a_2...a_n$ ikkilik sonning to‘g‘ri kodini hosil qilish uchun quyidagi ifodadan foydalanish mumkin:

$$A = \begin{cases} A, \text{ agar } A > 0 \text{ bo‘lsa,} \\ 1 - A, \text{ agar } A < 0 \text{ bo‘lsa.} \end{cases}$$

1.24-misol. $X_1 = +0,110011_2$ va $X_2 = -0,110011_2$ ikkilik kasr sonlarni to‘g‘ri kodda tasvirlang.

Javob: $[X_1]_{\text{to‘g‘}} = 0,110011$; $[X_2]_{\text{to‘g‘}} = 1,110011$.

EHMlarda to‘g‘ri kod oddiy bo‘lganligi uchun keng tarqalgan. Unda sonlarni xotirada saqlash, sonlarni ko‘paytirish katta qulayliklarga ega, lekin qo‘shish amalini to‘g‘ri kodda bajarish qiyinchilik tug‘diradi. Qo‘shish amalini murakkabroq kodlashtirish bilan bajarish mumkin.

Qo‘shimcha kod. Ikkilik manfiy sonning qo‘shimcha kodi quyidagi qoida bilan topiladi: sonning ishorali razryadiga bir yoziladi, boshqa hamma razryadlarda raqamlar o‘zaro teskari raqamlarga almashtiriladi, shundan keyin sonning eng kichik razryadiga bir qo‘shiladi.

Qo‘shimcha kod quyidagi ifoda bilan aniqlanadi:

$$A = \begin{cases} A, \text{ agar } A > 0 \text{ bo‘lsa,} \\ 10 + A, \text{ agar } A < 0 \text{ bo‘lsa.} \end{cases}$$

Yuqoridagi ifodadan ma’lumki, musbat ikkilik sonning qo‘shimcha kodi uning to‘g‘ri kodiga teng.

1.25-misol. $X_1 = +0,110011_2$ va $X_2 = -0,110011_2$ ikkilik kasr sonlarni qo‘shimcha kodda tasvirlang.

Javob: $[X_1]_{\text{qo‘sh}} = 0,110011$;

$[X_2]_{\text{qo‘sh}} = 10 + (0,110011) = 1,010110$.

Manfiy sonning qo‘shimcha kodidan uning to‘g‘ri kodiga o‘tish qiyin emas. Lekin amaliyotda boshqa qoidadan ham foydalanish mumkin. Sonning ishorali razryadiga bir yoziladi, qolgan razryadlaridagi sonlar o‘zaro teskari sonlarga almashtiriladi, keyin sonning eng kichik razryadiga bir qo‘shiladi.

1.26-misol. $X_1 = -0{,}0101_2$ ikkilik kasr sonni qo‘shimcha kodda tasvirlang.

Javob: $[X_1]_{qo‘sh} = 1{,}1010 + 0{,}0001 = 1{,}1011$.

Teskari kod. Teskari kod quyidagi ifoda bilan aniqlanadi:

$$A_{tes} = \begin{cases} A, \text{ agar } A > 0 \text{ bo‘lsa,} \\ 10 + A - 10^n, \text{ agar } A < 0 \text{ bo‘lsa.} \end{cases}$$

Manfiy ikkilik sonning teskari kodini quyidagi qoida asosida topish mumkin: sonning ishorali razryadiga bir yoziladi, hamma boshqa razryadlarda raqamlar o‘zaro teskari raqamlarga almashtiriladi.

1.27-misol. $X_1 = +0{,}110011_2$ va $X_2 = -0{,}110011_2$ ikkilik kasr sonlarni teskari kodda tasvirlang.

Javob: $[X_1]_{tes} = 0{,}110011$; $[X_2]_{tes} = 1{,}001100$.

Nazorat savollari va topshiriqlari

1. Sonlarni tasvirlashning qanday usullari mavjud?
2. Sonlarni qo‘zg‘almas vergulli tasvirlashni qanday tushunasiz?
3. Sonlarni qo‘zg‘aluvchi vergulli tasvirlashni qanday tushunasiz?
4. Sonlarni qo‘zg‘aluvchi vergulli tasvirlash nimaga asoslanadi?
5. Sonning normal shakli qanday bo‘ladi?
6. To‘g‘ri kodni ta’riflang.
7. Teskari kodni ta’riflang.
8. Qo‘shimcha kodni ta’riflang.

1.4. MASHINA KODLARIDAGI IKKILIK SONLARNI ALGEBRAIK QO‘SHISH QOIDALARI. QO‘ZG‘ALMAS VERGULLI SONLARNI QO‘SHISH

Qo‘zg‘almas vergulli sonlarni mashinalarda algebraik qo‘shish uchun to‘g‘ri, qo‘shimcha yoki teskari mashina kodlarining biridan foydalaniladi. Natija ham shu kodlardan birida olinadi. Ko‘p hollarda qo‘shimcha va teskari kodlar ishlatiladi, bunda ishorali razryad va kasr qism bir butun son deb qaraladi va mashina ularning to‘g‘ri kasr yoki butun ko‘rinishida berilganiga qaramay, xuddi musbat sonlardek, operatsiyalarni (amallarni) amalga oshiradi. Teskari va qo‘shimcha kodlarning ustunligi shundaki, yig‘indining

7-jadval

Oddiy ko‘rinishi	Kod		
	To‘g‘ri	Qo‘shimcha	Teskari
1-holat			
X_1=0,10101 + X_2=0,00101 X_3=0,11010	$[X_1]_{to‘g‘}$=0,10101 + $[X_2]_{to‘g‘}$=0,00101 $[X_3]_{to‘g‘}$=0,11010	$[X_1]_{qo‘sh}$=0,10101 + $[X_2]_{qo‘sh}$=0,00101 $[X_3]_{qo‘sh}$=0,11010	$[X_1]_{tes}$=0,10101 + $[X_2]_{tes}$=0,00101 $[X_3]_{tes}$=0,11010
2- holat			
X_1 = 0,10101 + X_2= –0,00101 X_3 = 0,10000	$[X_1]_{to‘g‘}$=0,10101 + $[X_2]_{to‘g‘}$=1,00101 $[X_3]_{to‘g‘}$=0,10000 (ayirish bajariladi)	$[X_1]_{qo‘sh}$=0,10101 + $[X_2]_{qo‘sh}$=1,11011 $[X_3]_{qo‘sh}$=10,11010 olib tashlanadi	$[X_1]_{tes}$=0,10101 + $[X_2]_{tes}$=1,11010 10,01111 +1 Siklik ko‘chirish $[X_3]_{tes}$=0,10000
3- holat			
X_1 = –0,10101 + X_2 = 0,00101 X_3 = –0,10000	$[X_1]_{to‘g‘}$=0,10101 + $[X_2]_{to‘g‘}$=0,00101 $[X_3]_{to‘g‘}$=1,10000 (ayirish bajariladi)	$[X_1]_{qo‘sh}$=1,01011 + $[X_2]_{qo‘sh}$=0,00101 $[X_3]_{qo‘sh}$=0,10000 Kodni almashtirish $[X_3]_{to‘g‘}$=1,10000	$[X_1]_{tes}$=1,01010 + $[X_2]_{tes}$=0,00101 $[X_3]_{tes}$=0,10000 Kodni almashtirish $[X_3]_{to‘g‘}$=1,10000
4- holat			
X_1 = –0,10101 + X_2 = –0,0101 X_3 = –0,11010	$[X_1]_{to‘g‘}$=1,10101 + $[X_2]_{to‘g‘}$=1,00101 $[X_3]_{to‘g‘}$=1,11010	$[X_1]_{qo‘sh}$=1,01011 + $[X_2]_{qo‘sh}$=0,11011 $[X_3]_{qo‘sh}$=11,00110 olib tashlanadi Kodni almashtirish $[X_3]_{qo‘sh}$=1,11010	$[X_1]_{tes}$=0,01010 + $[X_2]_{tes}$=1,11010 $[X_3]_{tes}$=11,00100 +1 Siklik ko‘chirish $[X_3]_{tes}$=1,00101 Kodni almashtirish $[X_3]_{to‘g‘}$=1,11010

Olingan natija bu ko‘rilayotgan holda natija ishorasi avtomatik holda hosil bo‘lishini yana bir bor tasdiqlaydi.

2-holat. Bunda musbat qo‘shiluvchi X_1 ning moduli X_2 ning modulidan katta bo‘lgani uchun

$$[X_1]_{qo‘sh} + [X_2]_{qo‘sh} < 1 \qquad (1.5)$$

munosabat o‘rinlidir. Bu holatda ishora raqami 1 ga teng. (1.5) ifodaning haqiqiyligi

$$|[X_1]_{qo'sh} + [X_2]_{qo'sh}| = 1 \quad va \quad |[X_1]_{qo'sh}| = |X_1| > |X_2|$$

dan kelib chiqadi. Shunday qilib, natija ishorasi quyidagi qismlardan tuziladi:

1-sonning ishora raqami	0,...
+	
2-sonning ishora raqami	1,...
+	
sonli razryadlardan ko‘chirma raqam	1
yig‘indining ishora raqami	10,...

tashlab yuboriladi

3-holat. Bunda manfiy qo‘shiluvchining moduli katta bo‘lgani uchun

$$|[X_1]_{qo'sh} + [X_2]_{qo'sh}| < 1 \qquad (1.6)$$

ifoda o‘rinli bo‘ladi. Bu ishora razryadiga ko‘chirma son doim nolga teng bo‘lishini bildiradi. U odatdagi ushbu munosabatlardan kelib chiqadi:

$$|[X_1]_{qo'sh} + [X_2]_{qo'sh}| = 1 \quad va \quad |[X_1]_{qo'sh}| = |X_1| < |X_2|.$$

Natijada yig‘indining ishorasi quyidagi yig‘indilardan iborat:

1-sonning ishora raqami	1,...
+	
2-sonning ishora raqami	0,...
+	
sonli razryadlardan ko‘chirma raqam	0
yig‘indining ishora raqami	1,...

Bu holatda ham ishoraning avtomatik kelib chiqishi isbotlandi.

4-holat. Razryad to‘ri to‘lib-toshmaganda va masshtab to‘g‘ri tanlanganda ishora razryadi albatta birga teng bo‘lishi kerak. Bu $|X_1| + |X_2| < 1$ munosabatdan kelib chiqadi. Bundan esa quyidagi ifoda kelib chiqadi:

$$|[X_1]_{qo'sh} + [X_2]_{qo'sh}| > 1. \qquad (1.7)$$

Bu holatda ishora paydo bo‘lishini tahlil qilib, quyidagini hosil qilamiz:

1-sonning ishora raqami		1,...
	+	
2-sonning ishora raqami		1,...
	+	
sonli razryadlardan ko‘chirma raqam		1
yig‘indining ishora raqami		11,...

tashlab yuboriladi

Shunday qilib, to‘rtala holatda ham qo‘shiluvchilarni oddiy qo‘shish orqali yig‘indining ishorasi avtomatik hosil bo‘lishini isbotladik. Bunday usullarda musbat ishora 1 bilan, manfiy ishora 0 bilan belgilanadi.

Amaliyotda ko‘pincha modifikatsiyalangan mashina kodlaridan foydalaniladi. Modifikatsiyalangan kodlarda sonlarning ishoralari ikkita raqam bilan belgilanadi: musbat ishora ikkita nol bilan va manfiy ishora ikkita bir bilan. Boshqa kombinatsiyalar (01 va 10) man etiladi va yig‘indining ishorali razryadlarida ularning paydo bo‘lishi razryad to‘rining to‘lib-toshishini anglatadi.

Nazorat savollari va topshiriqlari

1. Qo‘zg‘almas vergulli ikkilik sonlarni qo‘shish amalining 1-holati.
2. Qo‘zg‘almas vergulli ikkilik sonlarni qo‘shish amalining 2-holati.
3. Qo‘zg‘almas vergulli ikkilik sonlarni qo‘shish amalining 3-holati.
4. Qo‘zg‘almas vergulli ikkilik sonlarni qo‘shish amalining 4-holati.

1.5. QO‘ZG‘ALUVCHI VERGULLI IKKILIK SONLARNI QO‘SHISH

Agar normal formadagi ikkita $X_1 = m_1 \cdot 10^{P_1}$ va $X_2 = m_2 \cdot 10^{P_2}$ sonlar berilgan bo‘lsa, ularni qo‘shish uchun dastlab sonlarni bir xil tartibga keltirish, ya’ni qo‘shiluvchilardan biriga, masalan, birinchisiga quyidagi o‘zgartirishlarni kiritish kerak:

$$X_1 = m_1 \cdot 10^{P_1} = m_1^* \cdot 10^{P_2} = m_1^* \cdot 10^{P_{\text{um}}}.$$

Keyingi bosqichda asos darajasini qavsdan tashqariga chiqarib, mantissalarni qo‘shish lozim:

$$X_1 + X_2 = m_1^* \cdot 10^{P_{\text{um}}} + m_2 \cdot 10^{P_{\text{um}}} = (m_1^* + m_2)10^{P_{\text{um}}}.$$

O‘zgartirishda doim kichik sonni tanlash kerak, aks holda mantissalar yig‘indisi razryad to‘rining to‘lib-toshishiga olib keladi.

Shunday qilib, normal formadagi sonlarni mashinalarda qo‘shish amali 4 bosqichga bo‘linadi:

1) *Qo‘shiluvchilar tartibini tenglashtirish*. Kichik tartib kattasiga o‘zgartiriladi, o‘zgartirilgan son mantissasi esa mos razryadlar soniga o‘ngga suriladi. Amaliyotda qo‘shiluvchilar tartibi ayiriladi.

2) *Mantissalarni qo‘shish*. Qo‘shiluvchilar mantissalari modifikatsiyalangan kodlarning bittasiga o‘zgartiriladi. Qo‘shiluvchilar mantissalari qo‘zg‘almas vergulli kasr sonlardek qo‘shiladi. Sonlarning yig‘indisi mantissalarning yig‘indisiga teng bo‘ladi. Yig‘indining tartibi qo‘shiluvchilarning umumiy tartibiga, ya’ni katta sonning tartibiga teng. Zarur bo‘lsa, yig‘indining mantissasi to‘g‘ri kodga o‘tkaziladi.

3) *Natijani normallash*. Qo‘shiluvchilar mantissalarining qiymatlariga qarab yig‘indi: normallangan, o‘ngga denormallangan, chapga denormallangan bo‘lishi mumkin.

Agar qo‘shiluvchilar mantissalarining absolut qiymatlari birdan kichik bo‘lsa, razryad to‘ri to‘lib-toshishi mumkin, bu normallashning verguldan chap tomoniga buzilishini bildiradi. Bunday buzilishning belgisi yig‘indi modifikatsiyalangan kodining ishorali razryadlarida har xil raqamlarning paydo bo‘lishidadir (masalan, 01, 101,... yoki 10, 101, ...). Normallashni qayta tiklash uchun yig‘indining mantissasi o‘ngga bir xonaga suriladi va yig‘indining tartibi birga kamaytiriladi. Sonni bunday to‘g‘rilash *o‘ngga normallash* deyiladi.

Mantissalarning algebraik qo‘shilishida natija 0, 1, ... kam chiqishi mumkin. Bu normallashning verguldan o‘ng tomoniga buzilishini bildiradi. Bunday buzilishning belgisi yig‘indining mantissasi verguldan o‘ng tomonda bitta yoki bir nechta nolning paydo bo‘lishi bilan tavsiflanadi. Chapga normallashda mantissaning katta (birinchi) razryadi nolga teng bo‘lmasligi uchun uni kerakli razryadlar sonicha o‘ngga surish kerak. Yig‘indining tartibi kerakli birliklar soniga kamaytiriladi.

Agar yig‘indi mantissasining hamma razryadlari nolga teng bo‘lsa, normallash jarayoni o‘tkazilmaydi.

1.28-misol. Sonlarning yig‘indisini toping:

$A = +0{,}10100 \cdot 10^{+101}$ va $B = -0{,}10110 \cdot 10^{+100}$.

Yechish: 1) qo‘shiluvchilarning tartiblarini tenglashtirish uchun A sonning tartibidan B sonning tartibini ayirish kerak. Ayirishni modifikatsiyalangan qo‘shimcha kodda qo‘shish amali bilan almashtiramiz:

$$\begin{array}{r} P^M_{A\,\text{qo'sh}} = 00{,}101 \\ + \\ P^M_{B\,\text{qo'sh}} = 11{,}100 \\ \hline (P_A + P_B)^M_{\text{qo'sh}} = 00{,}001. \end{array}$$

P_A tartib P_B tartibdan bir birlikka katta bo'lganligi sababli B sonning mantissasini bir xona o'ngga suramiz, ya'ni $M_{B_{\text{to'g'}}} =$ $= 1{,}101100 = 1{,}01011$;

2) mantissalarni modifikatsiyalangan qo'shimcha kodda qo'shib chiqamiz:

$$\begin{array}{r} M^M_{A\,\text{qo'sh}} = 00{,}10100 \\ + \\ M^M_{B\,\text{qo'sh}} = 11{,}10101 \\ \hline (M_A + M_B)^M_{\text{qo'sh}} = 00{,}01001 \end{array}$$

normallash buzilgan holat.

3) $0{,}01001 \cdot 10^{+101}$ ga teng natijani normallash uchun uning mantissasini bir xona chapga surish va tartibidan birni ayirish kerak. Normallangan natija:

$$A + B = 0{,}1001 \cdot 10^{+100}.$$

1.29-misol. $A = 0{,}10100 \cdot 10^{+011}$ va $B = 0{,}11100 \cdot 10^{+101}$ sonlarini qo'shib chiqing.

Yechish: 1) qo'shiluvchilarning tartiblarini tenglashtiramiz:

$$\begin{array}{r} P^M_{A\,\text{qo'sh}} = 00{,}001 \\ + \\ P^M_{B\,\text{qo'sh}} = 11{,}011 \\ \hline (P_A + P_B)^M_{\text{qo'sh}} = 11{,}110. \end{array}$$

P_A tartib P_B tartibdan bir birlikka katta bo'lganligi sababli B sonning mantissasi ikki xona o'ngga suriladi, ya'ni $M_{A_{\text{to'g'}}} =$ $= 0{,}10100 = 0{,}00101$;

2) mantissalarni modifikatsiyalangan qo'shimcha kodda qo'shib chiqamiz:

$$M^{M}_{A\,\text{qo'sh}} = 00{,}00101$$

$$+$$

$$M^{M}_{B\,\text{qo'sh}} = 11{,}11101$$

$$(M_A + M_B)^{M}_{\text{qo'sh}} = 01{,}00001$$

normallash buzilgan holat.

3) $A + B = 01{,}00001 \cdot 10^{+101}$ ga teng natijani normallash uchun uning mantissasini bir xona o‘ngga surish va tartibiga bir qo‘shish lozim. Normallangan natija:

$$A + B = 0{,}100001 \cdot 10^{+110}.$$

16 asosli qo‘zg‘aluvchi vergulli sonlar ustida qo‘shish amalini ko‘rib chiqamiz. Bunday sonlar uchun normallash amalini bajarish va tartiblarni tenglashtirishda mantissani 4 ga karrali razryad soniga surish talab etiladi. Qo‘zg‘aluvchi vergulli sonlarning formati 4 baytga teng deb shartli ravishda qabul qilamiz: sonning ishorasini va xarakteristikasini yozish uchun 1 bayt, mantissasini yozish uchun esa 3 bayt ishlatiladi.

1.30-misol. $A = (324)_{16}$ va $B = (-14A6)_{16}$ sonlarini qo‘shing.

Yechish: 1) $A = +0{,}324 \cdot 16^{+3}$, $B = -0{,}14A6 \cdot 16^{+4}$ normallangan sonlarning xarakteristikalarini topamiz. A va B sonlarning tartiblari $P_A = +11$, $P_B = +100$, demak, ularning xarakteristikalari

$$P^{x}_{A} = 11 + 1000000 = 1000011;$$

$$P^{x}_{B} = 100 + 1000000 = 1000100.$$

Sonlarning shartli formatidan foydalanib, A va B ni normallangan formada yozamiz:

	Ishora	Xarakteristika	Mantissa
$A \to 0$		100 0011	0011 0010 0100 0000 0000 0000
$B \to 1$		100 0100	0001 0100 1010 0110 0000 0000

2) xarakteristikalarni tenglashtiramiz. Buning uchun xarakteristikalarni modifikatsiyalangan qo‘shimcha kodda ayirib chiqamiz:

$$
\begin{array}{rl}
 & P^x_{A\,\text{qo'sh}} = 00{,}1000100 \\
+ & P^x_{B\,\text{qo'sh}} = 11{,}0111101 \\
\hline
 & (P^x_A + P^x_B)_{\text{qo'sh}} = 100{,}0000001
\end{array}
$$

yo‘qoladi

Xarakteristikalarning ayirmasi +1 ga teng, demak, bir o‘n oltilik razryadga A sonning mantissasini o‘ngga surish kerak:

	Ishora	Xarakteristika	Mantissa
$A \to 0$		100 0100	0000 0011 0010 0100 0000 0000

3) A va B sonlarning mantissalarini qo‘shimcha kodda qo‘shib chiqamiz:

	Ishora	Mantissa
	$M_{A\,\text{qo'sh}} =$	00 0000 0011 0010 0100 0000 0000
+	$M_{B\,\text{qo'sh}} =$	11. 1110 1011 0101 1010 0000 0000
	$(M_A + M_B)_{\text{qo'sh}} =$	11. 1110 1110 0111 1110 0000 0000

Razryad to‘ri to‘lib-toshmagan. Natijaning mantissasi manfiy chiqqanligi uchun uni to‘g‘ri kodga o‘tkazamiz:

$$(M_A + M_B)_{\text{to'g'}} = 11.0001\ 0001\ 1000\ 0010\ 0000\ 0000.$$

To‘g‘ri kodda mantissalarning yig‘indisi normallangan. A va B sonlarning normallangan qo‘shish natijasini yozamiz:

	Ishora	Xarakteristika	Mantissa
$A + B = 1.$		100 0100	0001 0001 1000 0010 0000 0000

yoki $(A + B)_{16} = -0{,}1182 \cdot 16^{+4}$.

1.31-misol. $A = (+10B3)_{16}$ va $B = (-C7)_{16}$ sonlarini qo‘shing.

Yechish: 1) $A = +0{,}10B3 \cdot 16^{+4}$ va $B = -C7 \cdot 16^{+2}$. Normallangan sonlarning xarakteristikalarini topamiz.

A va B sonlarning tartiblari $P_A = +100$, $P_B = +10$, demak, xarakteristikalari

$$P^x_A = 100 + 1000000 = 1000100;$$
$$P^x_B = 10 + 1000000 = 1000010.$$

Sonlarning shartli formatlaridan foydalanib, A va B ni normallangan shaklda yozamiz:

	Ishora	Xarakteristika	Mantissa
$A \to 0$		100 0100	0001 0000 1011 0011 0000 0000
$B \to 1$		100 0010	1100 0111 0000 0000 0000 0000

2) xarakteristikalarni tenglashtiramiz. Buning uchun xarakteristikalarni modifikatsiyalangan qo‘shimcha kodda ayirib chiqamiz:

$$\begin{array}{rl} & P^x_{A\,\text{qo‘sh}} = 00{,}1000100 \\ + & \\ & P^x_{B\,\text{qo‘sh}} = 11{,}0111110 \\ \hline (P^x_A + P^x_B)_{\text{qo‘sh}} = & 100{,}0000010 \end{array}$$

yo‘qoladi

Xarakteristikalarning ayirmasi 2 ga teng, demak, B sonning mantissasini ikkita o‘n oltilik razryadga o‘ngga surish kerak. B sonning mantissasini suramiz:

	Ishora	Xarakteristika	Mantissa
$B \to 1.$		100 0100	0000 0000 1100 0111 0000 0000

3) mantissalarni qo‘shib chiqamiz:

	Ishora	Mantissa
+ $M_{A\,\text{qo‘sh}}$	$\widehat{00}.$	0001 0000 1011 0011 0000 0000
$M_{B\,\text{qo‘sh}}$	11.	1111 1111 0001 1001 0000 0000
$(M_A + M_B)_{\text{qo‘sh}}$	100.	0000 1111 1110 1100 0000 0000

yo‘qoladi

Mantissalarning yig‘indisi musbat. Natijani yozamiz:

	Natijaning ishorasi	Xarakteristika	Mantissa
$A + B = 0$		100 0100	0000 1111 1110 1100 0000 0000

Natija normallanmagan

4) natijani normallashtiramiz. Buning uchun natijaning mantissasini bitta o‘n oltilik razryadga chapga suramiz va xarakteristikadan birni ayirib chiqamiz. Natijani normallangan formada yozamiz:

Ishora	Xarakteristika	Mantissa

$A + B \to 0.$ $\overline{100\ 0011}$ $\overline{1111\ 1110\ 1100\ 0000\ 0000\ 0000}$

yoki $A + B = +0$, АУС $\cdot\, 6^3 = (\text{АУС})_{16}$.

Nazorat savollari va topshiriqlari

1. Qo‘zg‘aluvchi vergulli sonlarni qo‘shish amalining bosqichlari.
2. Berilgan sonlarni qo‘shing:
   a) $A = -0{,}110011 \cdot 2^{+11}$ va $B = -0{,}100001 \cdot 2^{-10}$;
   b) $A = -0{,}101101 \cdot 2^{-101}$ va $B = 0{,}100101 \cdot 2^{-10}$;
   d) $A = -0{,}100010 \cdot 2^{-11}$ va $B = 0{,}111101 \cdot 2^{-111}$.
3. Berilgan sonlarni qo‘shing:
   a) $A = 0{,}527\text{B} \cdot 16^{+3}$ va $B = 0{,}497\text{E} \cdot 16^{-4}$;
   b) $A = 0{,}73\text{AC} \cdot 16^{-2}$ va $B = 0{,}9\text{DEF} \cdot 16^{5}$.

2- bob. HISOBLASH VA MIKROPROTSESSORLI TEXNIKANING MANTIQIY ASOSLARI

2.1. BUL ALGEBRASINING ASOSIY TUSHUNCHALARI

Mantiq algebrasining asosiy tushunchalari. Barcha hisoblash va mikroprotsessorli texnika qurilmalarining matematik asosini mantiq algebrasi qoidalari tashkil qiladi. Uning asoschisi M. Bul (1815−1864) bo‘lgani uchun *Bul algebrasi* yoki *mantiq algebrasi* deb ataladi.

Mantiq algebrasida amallar mantiqiy fikrlar ustida olib boriladi. *Fikr* deganda unga nisbatan haqiqat yoki yolg‘on qiymatlari bo‘ladigan ixtiyoriy munosabat tushuniladi. Fikrlar oddiy va murakkab bo‘lishi mumkin: oddiy fikr boshqa fikrlardan farq qilmaydi, murakkablari esa ikki yoki undan ortiq oddiy fikrlardan iborat bo‘ladi. Oddiy fikrlar *mantiqiy o‘zgaruvchilar,* murakkab fikrlar esa *mantiqiy funksiyalar* deb ataladi. Fikrlar faqat rostligi yoki yolg‘onligi bilan baholanadi. Agar u rost bo‘lsa 1, yolg‘on bo‘lsa 0 qiymatni oladi. Ikkita fikrning rostlik qiymati bir xil bo‘lsa, ular *ekvivalent fikrlar* deb ataladi.

Mantiq algebrasida mantiqiy o‘zgaruvchilar lotin alifbosining bosh harflari bilan belgilanadi. Masalan, $A = 1$ yozuvi A mantiqiy o‘zgaruvchining rostlik qiymati 1, $A = B$ esa A va B mantiqiy o‘zgaruvchilar ekvivalentligini bildiradi. Masalan, A va B mantiqiy fikrlar berilgan bo‘lsin:

A = «Yer yassi»,

B = «Avtomobillarda dvigatel mavjud».

Bu mantiqiy fikrlar asosida $A = 0$, $B = 1$ deb yozish mumkin, chunki A mantiqiy fikr yolg‘on, B mantiqiy fikr esa rostdir.

EHMda mantiqiy o‘zgaruvchilarni tasvirlash uchun ikki holatli elektron elementlar ishlatiladi. Ixtiyoriy mantiqiy yoki Bul funksiya deb ataluvchi $X = f(A, B, C,..., N)$ funksiya ham 0 yoki 1 qiymatni qabul qilishi mumkin. Mantiqiy funksiyaning qiymati $A, B, C, ..., N$ o‘zgaruvchilarga bog‘liq. EHMlarning mantiqiy sxemalari, odatda, analitik ko‘rinishda yozilgan mantiqiy funksiya asosida quriladi. Mantiqiy funksiyaning ko‘rsatuvchi formasi shu funksiya tashkil topgan o‘zgaruvchilarning mumkin bo‘lgan mantiqiy munosabatlaridan tuzilgan rostlik jadvali hisoblanadi.

Mantiq algebrasi amallari. X mantiqiy funksiyaning tuzilishi $A, B, C, ..., N$ o‘zgaruvchilardan tashkil topgan va asosiy mantiqiy

amallar hisoblangan «HAMDA», «YOKI», «EMAS» yordamida amalga oshiriladi. Mantiqiy amallarni bajaruvchi elektron sxemalar *mantiqiy elementlar* deyiladi.

EMAS amali (mantiqiy inkor, inversiya). *A* fikrning *inkori* deb, uning natijasi *A* yolg‘on bo‘lganda rost, *A* rost bo‘lganda yolg‘on chiquvchi amalga aytiladi (8.1-jadval). Inkor *A* fikrning ustiga chiziqcha chizish bilan belgilanadi: $X = \overline{A}$ va *X* bu *A* dan olingan inversiya, deb o‘qiladi. Inkor amalini ishlab chiquvchi elektron sxema *invertor* yoki *EMAS mantiqiy sxemasi* deyiladi. Uning kelishilgan grafik ko‘rinishi 2.1-*a* rasmda keltirilgan. EMAS elementining kirishida signal bo‘lmasa, chiqishida signal hosil bo‘ladi.

YOKI amali (mantiqiy qo‘shish, dizyunksiya). Bu ikkita *A* va *B* o‘zgaruvchi ustidagi amaldir. *X* natija ikki o‘zgaruvchining bittasi rost bo‘lganda rost bo‘ladi, qolgan hollarda natija yolg‘on bo‘ladi (8.2-jadval):

$$X = A \vee B \Leftrightarrow X = A + B.$$

YOKI amalini bajaruvchi elektron sxema *YOKI mantiqiy sxemasi, dizyunktor, yig‘uvchi sxema* deb ataladi. Uning kelishilgan grafik ko‘rinishi 2.1-*b* rasmda keltirilgan. YOKI elementining hech bo‘lmaganda bitta qiymati 1 bo‘lsa, natija albatta 1 ga teng bo‘ladi. YOKI amali ixtiyoriy sondagi mantiqiy o‘zgaruvchilar uchun ham o‘rinli hisoblanadi:

$$X = A \vee B \vee C \vee ... \vee N.$$

HAMDA amali (mantiqiy ko‘paytirish, konyunksiya). Bu amal ham ikkita mantiqiy o‘zgaruvchi ustida boradigan mantiqiy jarayondir. Uning kelishilgan grafik ko‘rinishi 2.1-*d* rasmda keltirilgan. Uning natijasi *X* har ikkala o‘zgaruvchi 1 ga teng bo‘lganda 1 qiymatni qabul qiladi, qolgan hollarda natija 0 ga teng bo‘ladi (8.3-jadval). HAMDA amali mantiqiy ko‘paytirishni bildiruvchi «$\wedge$» yoki « $\cdot$ » belgilari bilan belgilanadi:

$$X = A \wedge B \Leftrightarrow X = A \cdot B.$$

8.1-jadval

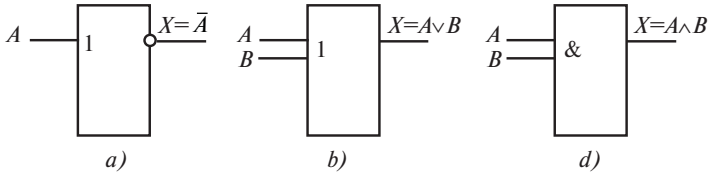
A	*X*
0	1
1	0

8.2-jadval

A	*B*	*X*
0	0	0
0	1	1
1	0	1
1	1	1

8.3-jadval

A	*B*	*X*
0	0	0
0	1	0
1	0	0
1	1	1



2.1-rasm. EMAS (*a*), YOKI (*b*), HAMDA (*d*) mantiqiy elementlarning shartli grafik ko‘rinishi.

Mantiq algebrasining asosiy qonunlari. Mantiq algebrasida 4 ta asosiy qonun mavjud.

1. O‘rin almashtirish yoki, mos ravishda, qo‘shish va ko‘paytirishning kommutativlik qonuni:

$$A \vee B = B \vee A; \tag{2.1}$$

$$A \wedge B = B \wedge A. \tag{2.2}$$

2. Taqsimot yoki, mos ravishda, qo‘shish va ko‘paytirishning assotsiativlik qonuni:

$$(A \vee B) \vee C = A \vee (B \vee C); \tag{2.3}$$

$$(AB) \vee C = A \vee (BC). \tag{2.4}$$

3. Inversiya qonuni yoki, mos ravishda, qo‘shish va ko‘paytirishning *de Morgan qoidasi:*

$$\overline{A \vee B} = \bar{A}\bar{B}; \tag{2.5}$$

$$\overline{AB} = \bar{A} \vee \bar{B}. \tag{2.6}$$

4. Qo‘shish va ko‘paytirishning, mos ravishda, distributivlik qonuni:

$$(A \vee B)C = AC \vee BC; \tag{2.7}$$

$$(AB) \vee C = (A \vee C)(B \vee C). \tag{2.8}$$

Bu qonunlarning haqiqat ekanligini ularni tavsiflaydigan murakkab mantiqiy ifodalarning rostlik jadvali yordamida isbotlash mumkin.

Mantiqiy ifodalarni o‘zgartirish uchun ushbu oson isbotlanuvchi ayniyatlardan foydalaniladi:

$$A \vee 0 = A; \qquad A \cdot 1 = A; \tag{2.9}$$

$$A \vee 1 = 1; \qquad A \cdot 0 = 0; \tag{2.10}$$

$$A \vee A = A; \qquad A \cdot A = A; \tag{2.11}$$

$$A \vee \bar{A} = 1; \qquad \bar{A} \cdot A = 0; \tag{2.12}$$

$$A = A. \tag{2.13}$$

Mantiq algebrasi qonunlari va ayniyatlari yordamida *yutilish qoidalari* nomini olgan ushbu munosabatlarni isbotlash mumkin:

$$A \vee AB = A; \quad A(A \vee B) = A. \tag{2.14}$$

Shuningdek, mazkur qonunlar va ayniyatlar asosida *yopishqoqlik munosabatlari* deb ataladigan ushbu munosabatlarni ham isbotlash imkoniyatlari mavjud:

$$AB \vee A\bar{B} = A;$$
$$(A \vee B)(A \vee \bar{B}) = A. \tag{2.15}$$

Bu qoidalar mantiqiy funksiyalarni soddalashtirishda keng qo‘llaniladi.

(2.5) va (2.6) de Morgan qoidalaridan quyidagilar kelib chiqadi:

$$\overline{A \vee B} = \bar{A}\bar{B}; \quad AB = \overline{\bar{A} \vee \bar{B}}. \tag{2.16}$$

Juftlik qonunlari (2.5) va (2.6) hamda ularning natijalari (2.16) ixtiyoriy sondagi o‘zgaruvchilar uchun ham o‘rinlidir:

$$\overline{(A \vee B \vee C \vee ... \vee N)} = \bar{A} \cdot \bar{B} \cdot \bar{C} \cdot ... \cdot \bar{N}; \tag{2.17}$$

$$\overline{(ABC...N)} = \bar{A} \vee \bar{B} \vee \cdot \bar{C} \vee ... \vee \bar{N}. \tag{2.18}$$

Mantiq algebrasining barcha qonun va ayniyatlari dizyunksiya uchun ham, konyunksiya uchun ham o‘rinlidir, bu esa mazkur amallarni ikki yoqlamalashtiradi.

Nazorat savollari va topshiriqlari

1. Mantiq algebrasining kommutativlik qonunini yozing.
2. Mantiq algebrasining assotsiativlik qonunini yozing.
3. Qo‘shish amali uchun de Morgan qoidasini yozing.
4. Ko‘paytirish amali uchun de Morgan qoidasini yozing.
5. Mantiq algebrasining distributivlik qonunini yozing.

2.2. MANTIQIY FUNKSIYALARNING TASVIRLANISHI

Mantiqiy funksiyalar jadval ko‘rinishida va analitik ko‘rinishda tasvirlanishi mumkin. Birinchi usul 8.1, 8.2, 8.3-jadvallarda ko‘rsatilgan. Bu jadvallarga o‘zgaruvchilarning mumkin bo‘lgan qiymatlariga funksiyaning mos qiymatlari qo‘yiladi (0 yoki 1). Bu usulni ixtiyoriy sondagi o‘zgaruvchilar uchun qo‘llash mumkin, lekin bunday yozuv ixcham hisoblanmaydi. Ma’lumki, funksiyani jadval usulida

aniqlovchi to‘plam elementlari soni 2^n ta (n – o‘zgaruvchilar soni) bo‘lib, n ning katta qiymatlari uchun jadval kattalashib ketadi.

Mantiqiy funksiyaning rostlik jadvalini tuzish, ko‘pincha, EHMlarning biror murakkab tizimni loyihalashidagi dastlabki bosqich bo‘lib hisoblanadi.

Mantiqiy funksiyalarni formulalar yordamida yozuvchi analitik usul oddiyligi bilan ajralib turadi. Amaliyotda mantiqiy funksiyalarning turli ko‘rinishdagi formalarini (shakllarini) farqlash mumkin.

Normal formalar. Bu formalar elementar konyunksiyalarning dizyunksiyalarini hamda elementar dizyunksiyalarning konyunksiyalarini ifodalaydi.

Elementar konyunksiyalar, bu konyunksiya bo‘lib, unda konyunktiv ravishda faqat alohida o‘zgaruvchilar bog‘lanadi. Masalan, quyidagilar elementar konyunksiyalardir:

$$\bar{A}\,BC,\ ABC,\ A\bar{B},\ AC,\ CD,\ ABCD.$$

Elementar dizyunksiyalar, bu dizyunksiya bo‘lib, unda dizyunktiv ravishda faqat alohida o‘zgaruvchilar bog‘lanadi. Masalan, quyidagilar elementar dizyunksiyalardir:

$$(\bar{A}\vee B),\ (\bar{A}\vee C),\ (A\vee B\vee\bar{C}),\ (\bar{A}\vee B\vee C).$$

Elementar konyunksiyalarning dizyunksiyasi ko‘rinishidagi normal forma *dizyunktiv normal forma (DNF)* deyiladi. Masalan,

$$X_{\mathrm{DNF}} = \bar{A}\bar{B}\vee AB.$$

Elementar dizyunksiyalarning konyunksiyasi ko‘rinishidagi normal forma *konyunktiv normal forma (KNF)* deyiladi. Masalan,

$$X_{\mathrm{KNF}} = (A\vee C)(A\vee\bar{B})(\bar{A}\vee B).$$

Mukammal normal formalar. Ixtiyoriy mantiqiy funksiya bir necha DNF va KNFga ega bo‘lishi mumkin. Mantiqiy funksiyalarni yagona shaklda ifodalash ularning mukammal normal formada yozish imkoniyatidan kelib chiqadi. Mantiqiy funksiyalarning mukammal normal formalarini mazkur funksiyalarning rostlik jadvali yordamida hosil qilish mumkin.

Mantiqiy funksiya yordamida ifodalangan *mukammal dizyunktiv normal forma (MDNF),* bu X funksiyaning qiymatlari 1 ga teng bo‘lgan konyunksiyalarining dizyunksiyasi ko‘rinishida yozilishidir. Bu dizyunksiyaning har bir alohida konyunksiyasi har bir o‘zgaruvchining to‘g‘ri yoki invers ko‘rinishini faqat bir marta qabul

qiladi. Bu konyunksiyalar o‘zgaruvchilar qiymatlarining aniq to‘plamida rostdir va *birlikning konstituentasi* yoki *minterm* nomi bilan yuritiladi.

Mantiqiy funksiyaning jadval ko‘rinishidan MDNF ko‘rinishidagi yozuvga o‘tish algoritmining mohiyati quyidagicha:

1) rostlik jadvalida X funksiya birga teng bo‘lgan qatorlar uchun mintermlar tuziladi. Agar qatordagi (A, B, C, ...) o‘zgaruvchining qiymati 0 ga teng bo‘lsa, mintermda shu o‘zgaruvchining inversiyasi yoziladi;

2) tuzilgan mintermlarning dizyunksiyasi yoziladi. U mantiqiy funksiyaning MDNFda ifodalanishidir.

Bu qoida *mantiqiy funksiyalarni birliklarda yozish qoidasi* deb ataladi. Unga asosan, mantiqiy funksiyaning (9-jadval) MDNFda yozilishi ushbu ko‘rinishga ega bo‘ladi:

$$X_{\text{MDNF}} = \bar{A}\bar{B}C \vee \bar{A}B\bar{C} \vee \bar{A}\bar{B}\bar{C} \vee ABC\,. \quad (2.19)$$

Mukammal konyunktiv normal forma (MKNF). Mantiqiy funksiya bilan ifodalangan MKNF, bu X funksiyaning qiymatlari 0 ga teng bo‘lgan dizyunksiyalarining konyunksiyasi ko‘rinishida yozilishidir. Bu konyunksiyaning har bir dizyunksiyasi har bir o‘zgaruvchining to‘g‘ri yoki invers ko‘rinishini faqat bir marta qabul qiladi. Bu dizyunksiyalar o‘zgaruvchilar qiymatlarining aniq to‘plamida nolga aylanadi va ularning nomi *nolning konstituentasi* yoki *maksterm* deb yuritiladi.

Mantiqiy funksiyaning jadval ko‘rinishidan MKNF ko‘rinishiga o‘tkazish algoritmining mohiyati quyidagicha:

1) rostlik jadvalidan X funksiyaning qiymati 0 ga teng qatorlar uchun makstermlar topiladi. Agar qatorda o‘zgaruvchining qiymati 1 ga teng bo‘lsa, uning invers qiymati yoziladi;

2) tuzilgan makstermlarning konyunksiyasi yoziladi. Bu esa mantiqiy funksiyaning MKNFda tasvirlanishi demakdir.

Ba’zi hollarda bu qoida *mantiqiy funksiyani nollar bo‘yicha yozish qoidasi* deyiladi. Masalan, mantiqiy funksiyaning (9-jadval) MKNFda yozilishi quyidagi ko‘rinishga ega bo‘ladi:

$$X_{\text{MKNF}} = (A \vee B \vee C)(A \vee \bar{B} \vee \bar{C})(A \vee \bar{B} \vee C). \quad (2.20)$$

(2.20) ga inversiya amalini qo‘llab, mantiqiy funksiya uchun MDNF va MKNF orasidagi bog‘liqlikni yozamiz:

$$X_{\text{MDNF}} = \bar{X}_{\text{MKNF}}. \quad (2.21)$$

N ta mantiqiy o‘zgaruvchi uchun mintermlar soni m va makstermlar soni M bir xil bo‘lib, u $m = M = 2^n$ ga teng. Lekin (2.19)

9-jadval

i- to‘plamning raqami	*A*	*B*	*C*	$X_i=f_i\ (A, B, C)$
0	0	0	0	0
1	0	0	1	1
2	0	1	0	1
3	0	1	1	0
4	1	0	0	1
5	1	0	1	0
6	1	1	0	0
7	1	1	1	1

va (2.20) da, mos ravishda, 4 ta minterm va 4 ta maksterm mavjud, chunki qolgan mintermlar 0 ga, makstermlar 1 ga tengdir.

Mantiqiy o‘zgaruvchilardagi mintermda yoki makstermda o‘zgaruvchilar soni *rang* deyiladi.

(2.19) va (2.20) ifodalarda mintermlar va makstermlarning rangi 4 ga teng bo‘ladi.

To‘liq aniqlanmagan mantiqiy funksiya, bu hech bo‘lmaganda bitta o‘zgaruvchilar to‘plamida qiymati aniqlanmagan funksiyadir. Masalan, 10-jadvalda keltirilgan rostlik jadvalida, ikkinchi o‘zgaruvchilar to‘plamida ($A=0$, $B=1$, $C=0$) qiymati ko‘rsatilmagan mantiqiy funksiya berilgan bo‘lsin. Bu ikkinchi to‘plamda funksiya aniqlanmaganligini bildiradi va u ixtiyoriy qiymat (0 yoki 1) qabul qilishi mumkin.

Ushbu ma’lumot asosida funksiyaga 0 yoki 1 qiymatni berish, qo‘yilgan masalaga qarab, funksiyani qayta ishlashning turli bosqichlarida amalga oshiriladi. Ba’zan funksiya MNDF va MKNF yozilayotganda aniqlanadi. Bu holda *X* funksiya to‘plami uchun 0 yoki 1 ni tanlashimizga qarab MKNF va MDNFning 2 ta variantini yozish mumkin:

1-holatda

$$X_{\text{MDNF1}} = \bar{A}\bar{B}C \vee \bar{A}BC \vee \bar{A}BC \vee A\bar{B}\bar{C} \vee ABC\ ;$$

$$X_{\text{MKNF1}} = (A \vee B \vee C)(\bar{A} \vee B \vee \bar{C})(\bar{A} \vee \bar{B} \vee C)\ ;$$

2-holatda

$$X_{\text{MDNF2}} = \bar{A}\bar{B}C \vee \bar{A}B\bar{C} \vee \bar{A}BC \vee A\bar{B}\bar{C} \vee ABC\ ;$$

$$X_{\text{MKNF1}} = (A \vee B \vee C)(A \vee \bar{B} \vee C)(\bar{A} \vee B \vee C)(\bar{A} \vee \bar{B} \vee C)\,.$$

10-jadval

i- to‘plamning raqami	*A*	*B*	*C*	$X_i = f_i(A, B, C)$
0	0	0	0	0
1	0	0	1	1
2	0	1	0	*
3	0	1	1	1
4	1	0	0	1
5	1	0	1	0
6	1	1	0	0
7	1	1	1	1

Mantiqiy funksiyalarning MDNF va MKNFlari EHM mantiqiy sxemasini analiz va sintez qilishda keng qo‘llaniladi. Bunday elementlarning ishlatilishi bir yoki bir nechta o‘zgaruvchilarning mantiqiy funksiyalaridan topiladi.

Nazorat savollari va topshiriqlari

1. Mantiqiy funksiyani birliklar bo‘yicha yozish qoidasi.
2. Mantiqiy funksiyani nollar bo‘yicha yozish qoidasi.
3. Mantiqiy funksiyaning tasvirlanishi (analitik formasi).
4. Mantiqiy funksiyaning tasvirlanishi (jadval formasi).

2.3. BITTA VA IKKITA O‘ZGARUVCHINING MANTIQIY FUNKSIYASI

Bitta o‘zgaruvchining mantiqiy funksiyasi. Bitta o‘zgaruvchining mantiqiy funksiyasi 11.1-jadvalda keltirilgan. Bunda faqat 2 ta funksiyaning *A* o‘zgaruvchiga bog‘liq emasligi ko‘rinib turibdi.

Ikkita o‘zgaruvchining mantiqiy funksiyasi. Ikkita o‘zgaruvchining mantiqiy funksiyalari 11.2 va 11.3-jadvallarda keltirilgan. Bunda faqat 10 ta funksiya *A* va *B* o‘zgaruvchilarga bog‘liqdir. Bu funksiyalardan ba’zilarini izohlaymiz.

Sheffer shtrixi faqat *A* va *B* rost bo‘lganda yolg‘on qiymat qabul qiluvchi $f_{14}(A, B)$ funksiyadir. Bu amalning belgisi sifatida «|» simvoli ishlatiladi. Sheffer funksiyasining kelishilgan ko‘rinishi:

$$f_{14}(A, B) = A \mid B,$$

bunda $f_{14}(A, B)$ funksiya aynan *A* va *B* bo‘lganligi noto‘g‘ri deb o‘qiladi.

11.1-jadval

$X=f(A)$	A		Shartli belgisi	Funksiyaning nomi
	0	1		
$X_0=f_0(A)$	0	0	0	0 konstanta
$X_1=f_1(A)$	0	1	A	A o‘zgaruvchi
$X_2=f_2(A)$	1	0	$\bar{A}$	A inkori
$X_3=f_3(A)$	1	1	1	1 konstanta

11.2-jadval

$X=f(A, B)$	A 0011 B 0101	Funksiya nomi	Shartli belgi-lanishi	Belgilar
$X_0=(A, B)$	0000	0 konstanta	0	
$X_1=f_1(A, B)$	0001	Mantiqiy ko‘paytirish, konyunksiya	AB	
$X_2=f_2(A, B)$	0010	B bo‘yicha taqiq etish funksiyasi	AB	
$X_3=f_3(A, B)$	0011	A o‘zgaruvchi	A	
$X_4=f_4(A, B)$	0100	A bo‘yicha taqiq etish funksiyasi	$B\Delta A$	Δ – taqiq etish
$X_5=f_5(A, B)$	0101	B o‘zgaruvchi	B	
$X_6=f_6(A, B)$	0110	Modul 2 bo‘yicha qo‘shish, mantiqiy teng-mas qiymatlilik	$A\oplus B$	$\oplus$ – tengmas qiymatlilik
$X_7=f_7(A, B)$	0111	Mantiqiy qo‘shish, dizyunksiya	$A\vee B$	$\vee$ – mantiqiy qo‘shish
$X_8=f_8(A, B)$	1000	Pirs amali	$A\downarrow B$	$\downarrow$ – Pirs amali
$X_9=f_9(A, B)$	1001	Mantiqiy tengqiymatlilik, ekvivalentlilik	$A\infty B$	∞ – ekviva-lentlilik
$X_{10}=f_{10}(A, B)$	1010	B inkori	$\bar{B}$	«–» – inver-siya, inkor
$X_{11}=f_{11}(A, B)$	1011	B dan A gacha implikatsiyasi	$B\rightarrow A$	
$X_{12}=f_{12}(A, B)$	1100	A inkori	$\bar{A}$	
$X_{13}=f_{13}(A, B)$	1101	A dan B gacha implikatsiyasi	$A\rightarrow B$	$\rightarrow$ – implikatsiya
$X_{14}=f_{14}(A, B)$	1110	Sheffer amali	$A \mid B$	$\rightarrow$ – Sheffer shtrixi
$X_{15}=f_{15}(A, B)$	1111	Konstanta	1	

11.3-jadval

$X=f(A, B)$	Funksiya nomi	Shartli belgila-nishi	$X=f(A, B)$ mantiqiy funksiyaning tasvirlanishi	
			MDNF	MKNF
$X_0=f_0(A, B)$	Nol konstanta	0	yo‘q	$(A\vee B)(\bar{A}\vee B)(A\vee\bar{B})\times(\bar{A}\vee\bar{B})$
$X_1=f_1(A, B)$	Mantiqiy ko‘paytirish, konyunksiya	AB	AB	$(\bar{A}\vee B)(A\vee\bar{B})(A\vee B)$
$X_2=f_2(A, B)$	B bo‘yicha taqiq etish funksiyasi	AB	AB	$(A\vee B)(A\vee\bar{B})(\bar{A}\vee\bar{B})$
$X_3=f_3(A, B)$	A o‘zgaruvchi	A	$AB\vee A\bar{B}$	$(A\vee B)(A\vee\bar{B})$
$X_4=f_4(A, B)$	A bo‘yicha taqiq etish funksiyasi	$B\Delta A$	AB	$(A\vee B)(\bar{A}\vee\bar{B})(\bar{A}\vee\bar{B})$
$X_5=f_5(A, B)$	B o‘zgaruvchi	B	$\bar{A}B\vee AB$	$(A\vee B)(\bar{A}\vee B)$
$X_6=f_6(A, B)$	Modul 2 bo‘yicha qo‘shish, mantiqiy tengmas qiymatlilik	$A\oplus B$	$\bar{A}B\vee A\bar{B}$	$(A\vee B)(\bar{A}\vee\bar{B})$
$X_7=f_7(A, B)$	Mantiqiy qo‘shish, dizyunksiya	$A\vee B$	$\bar{A}B\vee A\bar{B}\vee AB$	$(A\vee B)$
$X_8=f_8(A, B)$	Pirs amali	$A\downarrow B$	$\bar{A}\ \bar{B}$	$(\bar{A}\vee\bar{B})(\bar{A}\vee B)(A\vee\bar{B})$
$X_9=f_9(A, B)$	Mantiqiy tengqiymatlilik, ekvivalentlilik	$A\infty B$	$\bar{A}\ \bar{B}\vee AB$	$(\bar{A}\vee B)(A\vee\bar{B})$
$X_{10}=f_{10}(A,B)$	B inkori	$\bar{B}$	$\bar{A}\ \bar{B}\vee A\bar{B}$	$(A\vee\bar{B})(\bar{A}\vee\bar{B})$
$X_{11}=f_{11}(A,B)$	B dan A gacha implikatsiyasi	$B\to A$	$\bar{A}\ \bar{B}\vee A\bar{B}\vee AB$	$(A\vee\bar{B})$
$X_{12}=f_{12}(A,B)$	A inkori	$\bar{A}$	$\bar{A}\ \bar{B}\vee\bar{A}B$	$(\bar{A}\vee B)(\bar{A}\vee\bar{B})$
$X_{13}=f_{13}(A,B)$	A dan B gacha	$A\to B$	$\bar{A}\ \bar{B}\vee\bar{A}B\vee AB$	$(\bar{A}\vee B)$
$X_{14}=f_{14}(A,B)$	Sheffer amali	$A\|B$	$\overline{\bar{A}\bar{B}\vee\bar{A}B\vee A\bar{B}}$	$(\bar{A}\cdot\bar{B})$
$X_{15}=f_{15}(A,B)$	Bir konstanta	1	$\bar{A}\ \bar{B}\vee\bar{A}B\vee A\bar{B}\vee AB$	Yo‘q

MDNFda Sheffer funksiyasi quyidagi ko‘rinishga kelishi mumkin:

$$f_{14}(A,\ B) = \overline{A} \vee \overline{B} = \overline{AB},$$

ya’ni Sheffer amalining natijasi shu o‘zgaruvchilar konyunksiyasining inversiyasidir. Shuning uchun Sheffer amali HAMDA-EMAS amali deyiladi.

Pirs chizig‘i – faqat A va B yolg‘on bo‘lganda natijasi rost bo‘luvchi $f_8(A,\ B)$ funksiyadir. Bu amal belgisi sifatida «$\downarrow$» ishlatiladi. Pirs funksiyasining shartli ko‘rinishi quyidagicha:

$$f_8(A,\ B) = A \downarrow B,$$

bunda $f_8(A,\ B)$ funksiya A ham emas, B ham emas deb o‘qiladi. MDNFda Pirs funksiyasi quyidagi ko‘rinishga kelishi mumkin:

$$f_8(A,\ B) = \overline{A \vee B},$$

ya’ni Pirs amalining natijasi shu o‘zgaruvchilar dizyunksiyasining inkoridir. Shuning uchun Pirs amali YOKI-EMAS amali deyiladi.

Tengqiymatlilik yoki ekvivalentlilik funksiyasi faqat A va B ning qiymatlari teng bo‘lganda rost bo‘ladigan $f_9(A,\ B)$ funksiyadir. Tengqiymatlilik amali uchun «∞» qabul qilingan. Tengqiymatlilik funksiyasining shartli belgisi quyidagicha:

$$f_9(A,\ B) = A \infty B.$$

Bu funksiya quyidagicha o‘qiladi:

$f_9(A,\ B)$ funksiya A ham emas, B ham emas yoki A va B ga teng deb olinadi.

Tengmas qiymatlilik yoki *modul 2 bo‘yicha qo‘shish funksiyasi* faqat A va B ning har xil qiymatlarida rost bo‘luvchi $f_6(A,\ B)$ funksiyadir. Bu amal uchun «$\vee$» belgisi qabul qilingan, funksiyaning shartli ko‘rinishi:

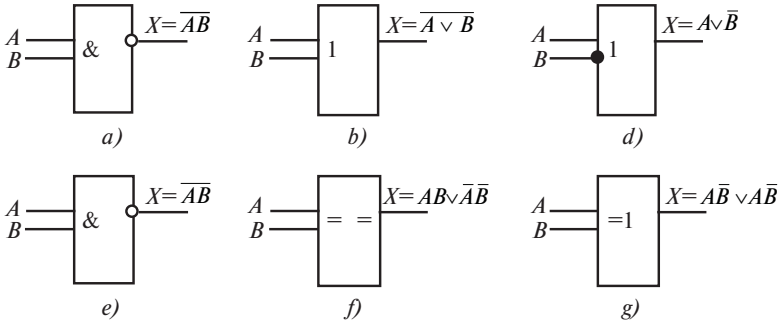
$$f_6(A,\ B) = A \vee B,$$

bunda $f_6(A,\ B)$ funksiya A yoki B ga teng, lekin bir vaqtda ikkalasiga ham teng emas, deb o‘qiladi.

Taqiq etish funksiyalari. $f_2(A,\ B)$ va $f_4(A,\ B)$ funksiyalar *taqiq etish mantiqiy funksiyalari* deb aytiladi. $f_4(A,\ B)$ taqiq funksiyasi B rost, A esa yolg‘on bo‘lganda rost bo‘ladi. $f_2(A,\ B)$ taqiq funksiyasi B yolg‘on, A rost bo‘lganda rost bo‘ladi. Taqiq amali uchun Δ belgisi ishlatiladi.

Funksiyalarning shartli ko‘rinishi:

$$f_2(A,\ B) = A\Delta B \quad \text{va} \quad f_4(A,\ B) = B\Delta A.$$



2.2-rasm.

Bu funksiyalar, mos ravishda, B emas, lekin A va A emas, lekin B deb o‘qiladi.

Implikatsiya funksiyalari. $f_{11}(A, B)$ va $f_{13}(A, B)$ funksiyalar *implikatsiya funksiyalari* deyiladi.

$f_{11}(A, B)$ implikatsiya funksiyasi faqat A yolg‘on va B rost bo‘lganda va faqat shundagina yolg‘ondir.

$f_{13}(A, B)$ implikatsiya funksiyasi faqat A rost va B yolg‘on bo‘lganda va faqat shundagina yolg‘ondir.

Implikatsiya uchun «$\rightarrow$» belgisi qabul qilingan. Bu funksiyaning shartli ko‘rinishi quyidagicha:

$$f_{11}(A, B) = A \rightarrow B \text{ va } f_{13}(A, B) = A \rightarrow B,$$

bu yerda: $f_{11}(A, B)$ funksiya A yolg‘on va B rost bo‘lganda va faqat shundagina yolg‘on, $f_{13}(A, B)$ funksiya A rost va B yolg‘on bo‘lganda va faqat shundagina yolg‘on deb o‘qiladi.

2.2-rasmda Sheffer (*a*), Pirs (*b*), implikatsiya (*d*), taqiq etish (*e*), ekvivalentlilik (*f*), mantiqiy tengmas qiymatlilik (*g*) elementlarining shartli grafik ko‘rinishlari keltirilgan.

Nazorat savollari va topshiriqlari

1. 3 ta o‘zgaruvchi uchun Sheffer funksiyasini yozing.
2. 3 ta o‘zgaruvchi uchun Pirs funksiyasini yozing.
3. Tengqiymatlilik funksiyasining shartli ko‘rinishini yozing.
4. Tengmas qiymatlilik funksiyasining shartli ko‘rinishini yozing.
5. Taqiq funksiyalarining shartli ko‘rinishini yozing.

2.4. MANTIQIY FUNKSIYANING NORMAL FORMASIDAN MUKAMMAL NORMAL FORMASIGA O‘TISH

Mantiqiy funksiyaning normal formasidan uning mukammal normal formasiga o‘tish analitik yoki grafik usullar yordamida amalga oshiriladi.

Analitik usul. Normal formadan farqli o‘laroq, mukammal normal forma faqat maksimal r rangli konyunksiyani (MDNF) yoki dizyunksiyani (MKNF) o‘z ichiga oladi. Bu quyidagi qoidalar asosida o‘tish imkoniyatini ta’minlaydi.

Ixtiyoriy DNFdan r rangli MDNFga o‘tish uchun DNFga kiruvchi k rangli ($k < r$) konyunksiyalarni ketma-ket quyidagi mantiqiy ifodaga ko‘paytirish zarur:

$$(Y_i \cdot \bar{Y}_i),$$

bu yerda: $Y_i = A, B, C, \ldots, N$ — bu berilgan konyunksiyaga kirmaydigan o‘zgaruvchilardan biridir. Har bir konyunksiya uchun bunday almashinishlar soni ($r - R$) ga teng bo‘lishi zarur.

2.1-misol. DNFda berilgan $f_{\text{DNF}}(A, B, C) = AB \vee C$ mantiqiy funksiyani MDNFga aylantiring.

1. Mantiq algebrasining (2.1), (2.2), (2.7) qonunlaridan va (2.12) ayniyatidan foydalanib, bu funksiyaning konyunksiyalarini 3 rangli mintermlarga aylantiramiz:

$$AB(C \vee \bar{C}) = ABC \vee AB\bar{C};$$

$$C = C(A \vee \bar{A}) = (AC \vee \bar{A}C)(B \vee \bar{B}) = ABC \vee \bar{A}BC \vee A\bar{B}C \vee \bar{A}\bar{B}C.$$

2. Chiqqan mintermlarni dizyunksiya belgisi bilan bog‘laymiz va (2.11) ayniyatdan foydalanib, quyidagini yozamiz:

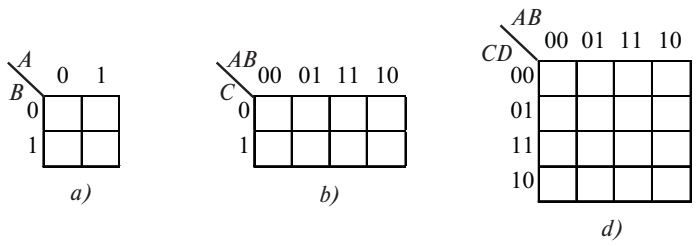
$$f_{\text{MDNF}} = ABC \vee AB\bar{C} \vee \overline{A\bar{B}}C \vee \bar{A}BC \vee \overline{A\bar{B}}C.$$

Ixtiyoriy KNFdan r rangli MKNFga o‘tish uchun KNFga kiruvchi k rangli ($k < r$) dizyunksiyasini quyidagi mantiqiy ifoda bilan ketma-ket qo‘shish kerak:

$$(Y_i\bar{Y}_i),$$

bu yerda: $Y_i = A, B, C, \ldots, N$ — dizyunksiyaga kirmaydigan o‘zgaruvchilardan biri. Har bir dizyunksiya uchun bunday almashinishlar soni ($r - R$) ga teng bo‘lishi zarur.

2.2-misol. KNFda berilgan $f_{\text{KNF}}(A, B, C) = A(B \vee \bar{C})$ mantiqiy funksiyani MKNFga aylantiring.



2.3-rasm. Karno kartasi 2 ta (*a*), 3 ta (*b*), 4 ta (*d*) o‘zgaruvchi uchun.

1. Mantiq algebrasining (2.3), (2.8) qonunlari va (2.9) ayniyatidan foydalanib, bu funksiyaning dizyunksiyalarini 3 rangli makstermlarga aylantiramiz:

$$A = A \vee B\bar{B} = (A \vee B)(A \vee \bar{B}) = (C \vee B \vee C\bar{C})(A \vee \bar{B} \vee C\bar{C}) =$$
$$= (A \vee B \vee C)(A \vee B \vee \bar{C})(A \vee \bar{B} \vee C)(A \vee \bar{B} \vee C);$$
$$B \vee \bar{C} = B \vee \bar{C} \vee B \vee A\bar{A} = (A \vee B \vee \bar{C})(\bar{A} \vee B \vee \bar{C}).$$

2. Hosil bo‘lgan makstermlarni konyunksiya belgisi bilan bog‘laymiz va (2.11) ayniyatdan foydalanib, quyidagini yozamiz:

$$f_{\text{MDNF}} = (A \vee B \vee C)(A \vee B \vee \bar{C})(A \vee \bar{B} \vee C)(A \vee \bar{B} \vee \bar{C})(\bar{A} \vee B \vee \bar{C}).$$

Grafik usul. Grafik usullarning eng oddiy va ko‘rgazmali ko‘rinishi bu Karno-Veych kartalari hisoblanadi. Karno kartasi – o‘zgaruvchilarning berilgan soni (n) uchun barcha (2^n) mintermlarni grafik tasvirlashdir. Har bir minterm katak ko‘rinishida tasvirlanadi va jadval tuzishda qo‘shni katakdagi mintermlar faqat bir o‘zgaruvchi bilan ajraladi. 2.3-rasmda Karno kartasining tasviri 2, 3, 4 ta o‘zgaruvchi bilan bog‘langan funksiya uchun berilgan. O‘zgaruvchilar kartaning chap burchagidan chiqqan diagonal chiziqning ikki tomonida joylashgan. O‘zgaruvchilarning qiymatlari kartaning tashqi tomonida ikkilik sonlar bilan yozilgan: 0 – o‘zgaruvchining invers qiymatiga, 1 – to‘g‘ri qiymatiga mos keladi. Bunday shart bilan Karno kartasining har bir katagiga mos keluvchi mintermni oson tasvirlash mumkin.

Karno kartalariga yana chetki kataklar yoki satrlar ham qo‘shni hisoblanadi, chunki ularda minterm bir o‘zgaruvchining qiymati bilan farqlanadi.

DNFda berilgan funksiyani MDNFga Karno kartasi orqali o‘tkazish algoritmi:

1. Berilgan mantiqiy funksiya uchun Karno kartasini tuzamiz.

2. Tarkibiga funksiyaning konyunksiyalari kirgan mintermlarni Karno kartasida 1 bilan belgilab qo‘yamiz.

3. 1 bilan belgilangan mintermlarni dizyunksiya belgilari bilan bog‘laymiz, bu berilgan mantiqiy funksiya uchun MDNF bo‘ladi.

2.3-misol. $f(A,\ B,\ C) = AB \vee C$ mantiqiy funksiyani Karno kartasi yordamida DNFdan MDNFga o‘tkazing.

Yechish: 1. Berilgan mantiqiy funksiya uchun Karno kartasini tuzamiz (2.4-rasm). Bu kartada AB konyunksiyani va C mantiqiy o‘zgaruvchini o‘z ichiga olgan mintermlarni 1 bilan belgilaymiz.

2. Mintermlarni dizyunksiya belgilari bilan bog‘lab, mantiqiy funksiyaning qiymatini MDNFda yozamiz:

$$f_{\text{MDNF}} = \bar{A}\bar{B}C \vee \bar{A}BC \vee A\bar{B}C \vee AB\bar{C} \vee ABC.$$

Mantiqiy funksiyani KNFdan MKNFga Karno kartasi yordamida o‘tkazish imkoniyatlari mavjud. Uni misolda ko‘rishimiz mumkin.

2.4-misol. KNFda berilgan mantiqiy funksiyani MKNFga o‘tkazing:

$$f_{\text{KNF}} = (A,\ B, C,\ D) = (A \vee B \vee C)(A \vee \bar{B} \vee D).$$

Yechish: 1. Berilgan funksiyadan invers qiymatini olamiz:

$$\overline{f_{\text{KNF}}(A,B,C,D)} = \bar{A}\bar{B}\bar{C} \vee \bar{A}B\bar{D}.$$

2. Hosil bo‘lgan mantiqiy funksiya uchun Karno kartasini tuzamiz (2.5-rasm). Bu kartada $\bar{A}\bar{B}\bar{C}$ va $\bar{A}B\bar{D}$ konyunksiyalarni va C mantiqiy o‘zgaruvchini o‘z ichiga olgan mintermlarni 1 bilan belgilaymiz.

3. Karno kartasidan foydalanib, bu funksiyaning MDNFda invers qiymatini yozamiz:

$$\overline{f_{\text{MKNF}}(A,B,C,D)} = \bar{A}\bar{B}\bar{C}\bar{D} \vee \bar{A}\bar{B}\bar{C}D \vee \bar{A}B\bar{C}D \vee \bar{A}BCD.$$

C \ AB	00	01	11	10
0	0	0	0	0
1	1	1	1	1

2.4-rasm. $f(A,\ B,\ C) = AB \vee C$ funksiya uchun Karno kartasi.

CD \ AB	00	01	11	10
00	1	1		
01	1			
11				
10		1		

2.5-rasm. $f_{\text{KNF}} = \bar{A}\bar{B}\bar{C} \vee \bar{A}B\bar{D}$ funksiya uchun Karno kartasi.

4. (2.13) ayniyatdan foydalanib, mantiqiy funksiyaning invers qiymatini yozamiz:

$$f_{\text{MKNF}} = (A \vee B \vee C \vee D)(A \vee B \vee C \vee \bar{D})(A \vee \bar{B} \vee C \vee D)(A \vee \bar{B} \vee \bar{C} \vee \bar{D}).$$

Hosil bo‘lgan funksiya berilgan mantiqiy funksiyaning MKNFda ko‘rinishini bildiradi.

Nazorat savollari va topshiriqlari

1. Berilgan mantiqiy funksiyalarni DNF va KNFga o‘tkazing:

$$X_1 = \overline{(AB \vee B\bar{C})}\,\overline{AB}; \quad X_2 = \overline{\overline{ABC} \vee \overline{B\bar{C}}} \vee A\bar{C}.$$

2. Berilgan mantiqiy funksiyalarni analitik usul bilan MDNFga o‘tkazing:

$$X_1 = A \vee \bar{B}C \vee \bar{A}BC; \quad X_2 = A \vee \bar{B}C.$$

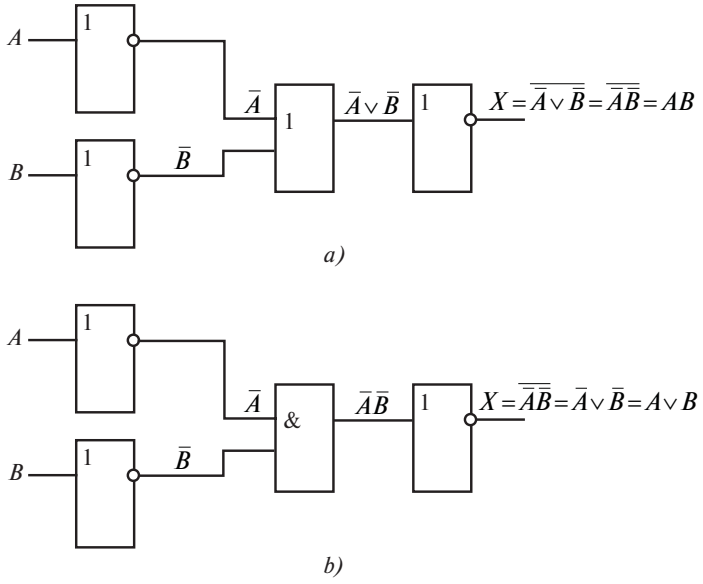
3. Berilgan mantiqiy funksiyalarni Karno kartalari yordamida MDNFga o‘tkazing:

$$X_1 = \bar{A}BC \vee A\bar{B}C \vee \bar{A}B\bar{C}; \quad X_2 = A\bar{B}\bar{C}D \vee \bar{A}D \vee \bar{B}C\bar{D}.$$

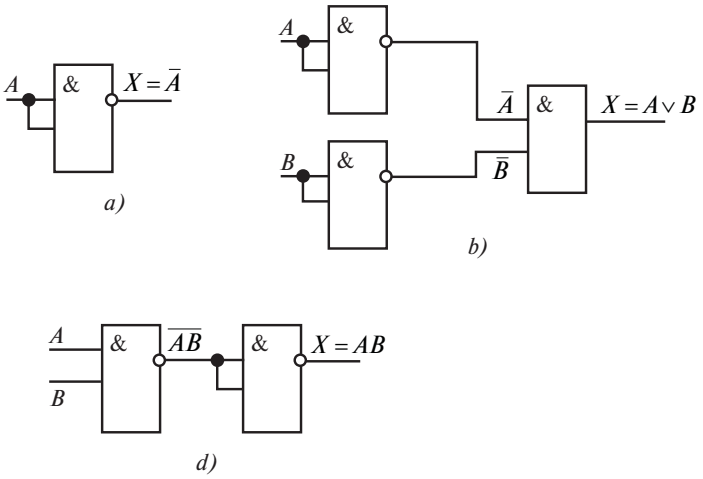
2.5. MANTIQ ALGEBRASI MANTIQIY FUNKSIYALARINING FUNKSIONAL TO‘LIQ TIZIMLARI

Mantiqiy funksiyalarning funksional to‘liq tizimi yoki bazisi deb mantiqiy funksiyalar X_1, X_2, X_3, ..., X_n tizimiga aytiladi. Bazis yordamida mantiq algebrasining ixtiyoriy funksiyasi izohlanadi. Funksional to‘liq tizim sifatida HAMDA, YOKI, EMAS (1-bazis); HAMDA, EMAS (2-bazis); YOKI, EMAS (3-bazis); HAMDA-EMAS yoki Sheffer bazisi (4-bazis); YOKI-EMAS yoki Pirs bazisi (5-bazis); HAMDA-YOKI-EMAS (6-bazis) bazislari olingan. HAMDA, YOKI, EMAS bazisi asosiy bazis deb qabul qilingan, chunki har qanday murakkab funksiyani MDNF va MKNF ko‘rinishida yozish mumkin. Bazislar ayriluvchi va minimal bo‘lishi mumkin.

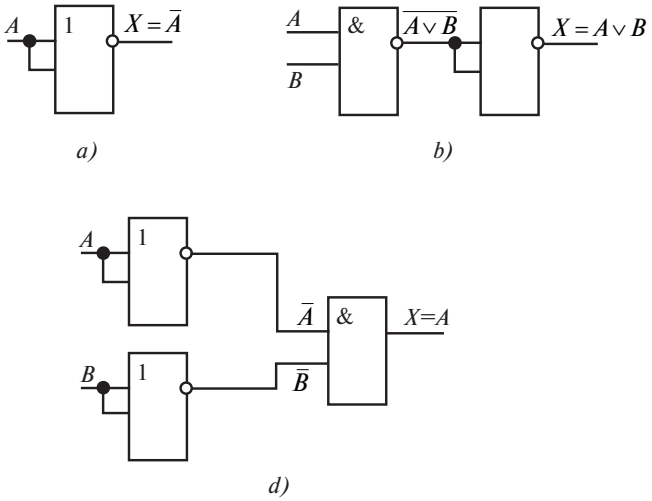
HAMDA, YOKI, EMAS bazisi ayriluvchi tizim hisoblanadi, chunki undan ba’zi funksiyalarni ayirish mumkin. Masalan, de Morgan qonunlaridan foydalanib, HAMDA funksiyasining o‘rniga YOKI va EMASni, YOKIning o‘rniga HAMDA bilan EMASni qo‘yish mumkin. 2.6-rasmda YOKI va EMAS elementlaridan tuzilgan HAMDA elementining va HAMDA bilan EMAS elementlaridan tuzilgan YOKI elementining grafik tuzilishi keltirilgan.



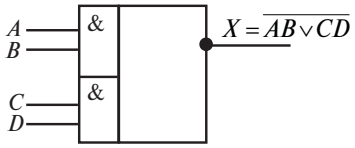
2.6-rasm. YOKI-EMAS bazisida HAMDA mantiqiy sxemasi (*a*); HAMDA-EMAS bazisida YOKI mantiqiy sxemasi (*b*).



2.7-rasm. Sheffer elementlaridan tuzilgan EMAS (*a*), HAMDA (*b*), YOKI (*d*) mantiqiy sxemalari.



2.8-rasm. Pirs elementlaridan tuzilgan EMAS (*a*), HAMDA (*b*), YOKI (*d*) mantiqiy sxemalari.



2.9-rasm. HAMDA-YOKI-EMAS elementining shartli grafik ko‘rinishi.

HAMDA, EMAS va YOKI, EMAS bazislari *normal* bazis deb hisoblanadi, chunki bu bazislardan hech bo‘lmaganda bitta funksiyani olib tashlashda butun tizim to‘liqsiz tizimga aylanib qoladi.

Sheffer elementlaridan tuzilgan EMAS, HAMDA, YOKI mantiqiy elementlarining strukturasi 2.7-rasmda keltirilgan.

Pirs elementlaridan tuzilgan EMAS, HAMDA, YOKI mantiqiy elementlarining strukturasi 2.8-rasmda keltirilgan.

EHM bloklari va qismlarini qurishda ko‘pincha HAMDA-YOKI-EMAS bazisi ishlatiladi (2.9-rasm).

Nazorat savollari va topshiriqlari

1. YOKI-EMAS bazisida HAMDA mantiqiy sxemasini chizing.
2. HAMDA-EMAS bazisida YOKI mantiqiy sxemasini chizing.
3. Sheffer elementlaridan tuzilgan EMAS mantiqiy sxemasini chizing.
4. Sheffer elementlaridan tuzilgan HAMDA mantiqiy sxemasini chizing.
5. Sheffer elementlaridan tuzilgan YOKI mantiqiy sxemasini chizing.
6. Pirs elementlaridan tuzilgan EMAS mantiqiy sxemasini chizing.
7. Pirs elementlaridan tuzilgan HAMDA mantiqiy sxemasini chizing.
8. Pirs elementlaridan tuzilgan YOKI mantiqiy sxemasini chizing.

2.6. MANTIQ ALGEBRASINING MANTIQIY FUNKSIYALARINI MINIMALLASHTIRISH

Mantiqiy funksiyaning minimal formasi deb boshqa hech qanday qisqartirishlarni qabul qilmaydigan formaga aytiladi.

Mantiqiy funksiyani minimal normal formada ta'kidlash *minimallashtirish* deyiladi. Minimallashtirishda qurilmalarning eng kam sarflanishi talabidan kelib chiqiladi, chunki har bir elementar mantiqiy funksiyaga ma'lum fizik element mos keladi. Mantiqiy funksiyani minimallashtirish uchun turli usullar ishlatiladi: Karnoning minimallashtiruvchi kartalari, Kvayn usuli, mantiq algebrasi qoida va munosabatlari yordamida ketma-ket o'chirish usuli va h.k.

O'zgaruvchilarni ketma-ket o'chirish usuli. O'zgaruvchilarni mantiq algebrasi munosabatlari va qonunlari yordamida ketma-ket o'chirish usuli minimallashning oddiy usuli hisoblanadi. Mantiqiy funksiyalarni har qanday soddalashtirish qo'shish natijasida alohida o'zgaruvchilarni yo'qotishga olib keladigan mintermlarning umumiy qo'shiluvchilarini qavsdan tashqariga chiqarish natijasida hosil bo'ladi. Ma'lumki, berilgan mintermdan biror o'zgaruvchini ajratish uchun unga faqat shu o'zgaruvchining qiymati bilan farqlanuvchi boshqa minterm qo'shish bilan bajariladi. Mintermlar juftliklarining o'zgaruvchi rangining pasayishi orqali boruvchi bunday tanlov *mintermlarning ulashishi* deyiladi.

2.5-misol. Berilgan mantiqiy funksiyani o'zgaruvchilarni ketma-ket o'chirish usuli bilan minimallashtiring:

$$X_{\text{MDNF}} = A\bar{B}\bar{C} \vee AB\bar{C} \vee ABC.$$

Yechish. Mintermlarni guruhlab hamda (2.1), (2.4), (2.8) va (2.12) dan foydalanib, berilgan mantiqiy funksiya uchun quyidagini yozamiz:

$$X = A\bar{B}\bar{C} \vee AB(C \vee \bar{C}) = A\bar{B}\bar{C} \vee AB = A(B \vee B)(\bar{B} \vee \bar{C}) = A(B \vee C).$$

Ko‘p sonli o‘zgaruvchilar bo‘lganda mintermlarni bir-biridan faqat 1 ta o‘zgaruvchining qiymati bilan farq qiluvchi guruhlarga ajratish ancha qiyin masaladir. Bundan tashqari, ba’zi mintermlar bir vaqtda bir necha guruhlarga kirishi va natijada bu guruhlar yechilmay qolishi ham mumkin. Mintermlarni har xil yo‘llar bilan guruhlab, berilgan funksiyaning turli qisqa formalarini olishimiz lozim, lekin olingan formalardan birini minimal deb aytolmaymiz. Bunday hollarda boshi berk yechim, ya’ni endi qisqarmaydigan minimal bo‘lmagan yechim hosil bo‘lishi mumkin.

Masalan,

$$X_{\text{MDNF}} = A\bar{B}C \vee \bar{A}\bar{B}C \vee \bar{A}\bar{B}\bar{C} \vee \bar{A}B\bar{C} \vee ABC \qquad (2.22)$$

mantiqiy funksiya uchun, mintermlarni guruhlab, quyidagini yozamiz:

$$X = (A\bar{B}C \vee \bar{A}\bar{B}C) \vee (A\bar{B}C \vee ABC) \vee (\bar{A}B\bar{C} \vee \bar{A}\bar{B}\bar{C}) \vee (A\bar{B}\bar{C} \vee AB\bar{C}).$$

(2.8) va (2.12) dan foydalanib, funksiyani quyidagi ko‘rinishda ifodalaymiz:

$$X = \bar{B}C(A \vee \bar{A}) \vee AC(B + \bar{B}) \vee \bar{A}\bar{C}(\bar{B} \vee \bar{B}) \vee B\bar{C}(A \vee \bar{A}) =$$
$$= B\bar{C} \vee AC \vee \bar{A}\bar{C} \vee B\bar{C}. \qquad (2.23)$$

Bu ifodadan ko‘rinib turibdiki, mazkur formani boshqa qisqartirib bo‘lmaydi, lekin u minimal ham emas. Bunday normal qisqartirilgan formaga kiruvchi konyunksiyalar *implikantlar* deyiladi.

Kvayn usuli. Kvayn usuli past rangli mantiqiy funksiyalar uchun qo‘llaniladi va boshlang‘ich mantiqiy funksiyalar albatta MDNFda berilishi kerak.

Mantiqiy funksiyalarning boshi berk hamda minimal formalarini o‘zaro taqqoslash maqsadida, Kvayn usulini (2.22) da ifodalangan funksiyaning mantiqiy ifodasi misolida ko‘rib chiqamiz.

Kvayn usuli bir necha bosqichdan iborat.

1-b o s q i c h . **Qisqartirilgan normal formani topish.** Bir-biridan faqat 1 ta o‘zgaruvchining qiymati bilan farqlanadigan mintermlar juftligini aniqlash uchun 12-jadval tuziladi. Bunday mintermlarning yig‘indisi bu jadvalga yoziladi.

Bog‘lash natijasida 1-bosqichning bu qadamida natijaviy qiymat 2 rangli oddiy implikantlarning dizyunksiyasini tavsiflaydi. Yutilish sodir bo‘lmaydigan mintermlar bu misolda istisno.

12-jadvalga asoslanib va (2.11) ni hisobga olib, natijaviy ifodani quyidagi ko‘rinishga keltiramiz:

$$X = B\bar{C} \vee \bar{A}\bar{B} \vee \bar{A}\bar{C} \vee \bar{B}C \vee AB \vee AC. \qquad (2.24)$$

12-jadval

Mintermlar	$A\bar{B}C$	$\bar{A}\bar{B}C$	$\bar{A}\bar{B}\bar{C}$	$\bar{A}B\bar{C}$	$AB\bar{C}$	ABC
$A\bar{B}C$	1	$\bar{B}C$				AC
$\bar{A}\bar{B}C$	$\bar{B}C$	1	$\bar{A}\bar{B}$			
$\bar{A}\bar{B}\bar{C}$		$\bar{A}\bar{B}$	1	$\bar{A}\bar{C}$		
$\bar{A}B\bar{C}$			$\bar{A}\bar{C}$	1	$B\bar{C}$	
$AB\bar{C}$				$B\bar{C}$	1	AB
ABC	AC				AB	1

13-jadval

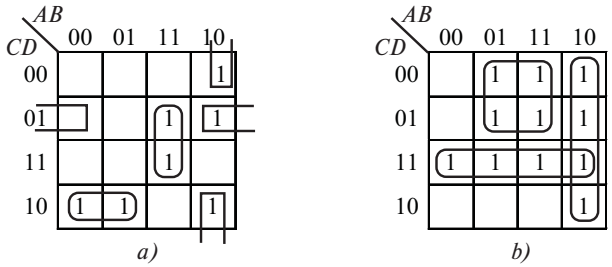
Mintermlar	$A\bar{B}C$	$\bar{A}\bar{B}C$	$\bar{A}\bar{B}\bar{C}$	$\bar{A}B\bar{C}$	$AB\bar{C}$	ABC
$\bar{B}C$	+	+				
$\bar{A}\bar{B}$		+	+			
$\bar{A}\bar{C}$			+	+		
$B\bar{C}$				+	+	
AB					+	+
AC	+					+

(2.24) ifodaga faqat 2 rangli implikantlar kirgan. Demak, ifodani yutilish amali bilan boshqa qisqartirib bo‘lmaydi. (2.24) ifoda berilgan funksiyaning qisqartirilgan normal formasi bo‘lib xizmat qiladi.

2-b o s q i c h . **Belgilarni o‘rnatish va minimal qoplagichni o‘rnatish.** Ushbu bosqichda 13-jadval tuziladi, unda satrlar soni (2.24) ifoda orqali topiladigan implikantlar soniga teng, ustunlarida esa berilgan mantiqiy funksiyaning chiqish ifodasiga kiruvchi hamma mintermlar joylashgan bo‘ladi. Belgilar, oddiy implikant berilgan mintermga kirsa, ustun va satrlarning kesishuvida qo‘yilgan.

Minimal formalar ko‘p bo‘lishi mumkin, lekin har bir mantiqiy funksiyaga faqat bitta qisqartirilgan normal forma mos keladi. Hamma minimal formalar 13-jadvaldan quyidagicha olinadi. Mantiqiy funksiyaning minimal formasi berilgan funksiyaning hamma mintermlarini qoplovchi implikantlarni o‘z ichiga olishi kerak.

13-jadvaldan hamma mintermlar berilgan funksiyaning $\bar{B}C$, AB, $\bar{A}\bar{C}$ yoki $\bar{A}\bar{B}$, $B\bar{C}$, AC implikantlari bilan qoplanganligi ko‘rinib turibdi:



2.10-rasm. Karno kartasida kataklar (maksterm yoki mintermlar)ning bog‘lanishi.

$$X_{\min 1} = B\bar{C} \vee \bar{A}\bar{C} \vee AB\,, \qquad (2.25)$$

$$X_{\min 2} = \bar{A}\bar{B} \vee B\bar{C} \vee AC\,. \qquad (2.26)$$

Demak, bu oldin olingan boshi berk forma (2.23) bilan ustma-ust tushmaydi. Bu ifodalarning teng kuchliligini berilgan tenglama o‘zgaruvchilariga ixtiyoriy qiymatlarni qo‘yib oson tekshirish mumkin. O‘zgaruvchilar soni 5 tadan ko‘p bo‘lsa, ularga Kvayn usulini qo‘llash maqsadga muvofiq emas.

Karnoning minimallashtiruvchi kartalari usuli. Karnoning minimallashtiruvchi kartalari usuli mantiqiy funksiyalarni minimallashtirish uchun keng qo‘llaniladi. Ularni Karno kartasi yordamida minimallashtirish asosini quyidagilar tashkil etadi: kartaning qo‘shni kataklarida joylashgan 2 ta minterm 1 ta o‘zgaruvchisi kam bo‘lgan bitta konyunksiya bilan almashtirilish mumkin. Agar ikki juft mintermlar qo‘shni bo‘lsa, 2 ta o‘zgaruvchisi kam bo‘lgan konyunksiya bilan almashtirilishi mumkin.

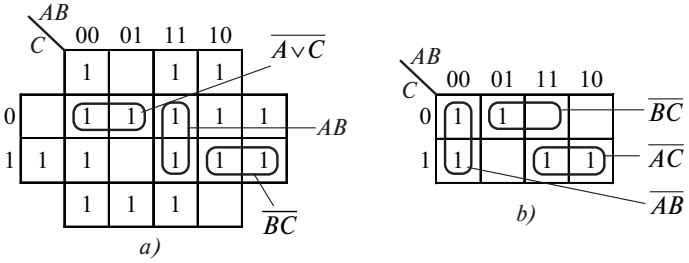
Umumiy holda, agar 2^n ta qo‘shni kataklarda mintermlar bo‘lsa, n ta o‘zgaruvchini yo‘qotish mumkin.

Minimallashtirishda qo‘shni kataklar faqat gorizontal va vertikal bo‘yicha yonma-yon joylashgan kataklargina emas, balki kartaning teskari tomonidagi kataklar ham hisoblanishini e’tiborga olish kerak: kataklar 2 tadan (2.10-*a* rasm), 4 tadan (2.10-*b* rasm) va h.k. birlashtirilishi, Karno kartasining bitta va o‘sha bir nechta guruhlariga kirishi mumkin.

MDNF va MKNFda berilgan mantiqiy funksiyalarni Karno kartalari yordami bilan minimallashtirish mumkin.

2.6-misol. Karno kartalari yordamida mantiqiy funksiyani minimallashtiring:

$$X_{\text{MDNF}} = A\bar{B}C \vee \bar{A}\bar{B}C \vee \bar{A}\bar{B}\bar{C} \vee \bar{A}B\bar{C} \vee AB\bar{C} \vee ABC.$$



2.11-rasm. Karno kartasi yordamida $X_{\text{MDNF}} = A\bar{B}C \vee \bar{A}\bar{B}C \vee \bar{A}\bar{B}\bar{C} \vee \bar{A}B\bar{C} \vee AB\bar{C} \vee ABC$ funksiyani minimallashtirish.

Yechish: 1. *A, B* va *C* o‘zgaruvchilar uchun Karno kartalarini tuzamiz (2.11-*a* rasm). Kartada 1 bilan $A\bar{B}C$, $\bar{A}\bar{B}C$, $\bar{A}\bar{B}\bar{C}$, $AB\bar{C}$, ABC, $\bar{A}B\bar{C}$ mintermlarni belgilab olamiz.

2. Karno kartasida har biri 2 ta mintermdan iborat mintermlar 3 ta guruhni tashkil etadi. Birinchi guruh $\bar{A}B\bar{C}$ va $\bar{A}\bar{B}\bar{C}$ dan iborat. (2.17) ayniyatdan foydalanib, ushbu guruhdan *B* o‘zgaruvchini o‘chirish mumkin.

Ikkinchi guruh *ABC* va $AB\bar{C}$ dan iborat. Bu guruhdan *C* o‘zgaruvchini o‘chirish mumkin.

Uchinchi guruh esa $AB\bar{C}$ va $A\bar{B}\bar{C}$ dan iborat. Undan *A* o‘zgaruvchini o‘chirish mumkin.

3. Minimallashtirilgan mantiqiy funksiyani DNFda yozamiz:

$$X_{\min 1} = AB \vee \bar{B}C \vee \bar{A}\bar{C}\,. \qquad (2.27)$$

Minterm guruhlarini boshqacha tanlab, (2.22) tenglama bilan berilgan mantiqiy funksiyaning ikkinchi minimal formasini topamiz (2.11-*b* rasm):

$$X_{\min 2} = \bar{A}\bar{B} \vee \bar{B}C \vee AC\,. \qquad (2.28)$$

Hosil qilingan (2.27) va (2.28) ifodalar (2.25) va (2.26) ifodalar bilan muvofiq keladi.

Nazorat savollari va topshiriqlari

1. O‘zgaruvchilarni ketma-ket o‘chirish usuli.
2. Kvayn usuli.
3. Karnoning minimallashtiruvchi kartalari usuli.
4. Berilgan mantiqiy funksiyani Karno kartalari yordamida minimallashtiring.

2.7. MANTIQIY FUNKSIYALARNI UNIVERSAL BAZISLARDA YOZISH

Mantiqiy funksiyalarni HAMDA-EMAS va YOKI-EMAS universal bazislarida yozish quyidagicha amalga oshiriladi.

1. Berilgan mantiqiy funksiya HAMDA, YOKI, EMAS bazisida minimallashtiriladi.

2. Olingan natija ustiga 2 marta inkor o‘rnatiladi va (2.16) ifoda yordamida universal HAMDA-EMAS yoki YOKI-EMAS bazislariga o‘tkaziladi.

3. Mantiqiy funksiyani almashtirishda quyidagi ifodalar bajariladi:

a) HAMDA-EMAS bazisida:

$$A\bar{B} = A(A\bar{B}),$$

$$(\bar{A}B)\vee(A\bar{B}) = \overline{[\overline{A(\overline{AB})}][\overline{A(\overline{AB})}]}, \qquad (2.29)$$

$$\bar{A} = \overline{A\cdot 1},$$

$$\bar{A} = \overline{A\cdot A};$$

b) YOKI-EMAS bazisida:

$$A\vee\bar{B} = A\vee(\overline{A\vee B}),$$

$$(A\vee\bar{B})(\bar{A}\vee B) = \overline{A\vee(\overline{A\vee B})}\vee\overline{B\vee(\overline{A\vee B})}, \qquad (2.29)$$

$$\bar{A} = \overline{A\vee 0},$$

$$\bar{A} = \overline{A\vee A}.$$

Funksional sxemalarni Sheffer elementlari yordamida tuzishda mantiqiy funksiya minimal DNF ko‘rinishida tasvirlanadi. Funksional sxemalarni Pirs elementlari asosida tuzishda esa mantiqiy funksiya minimal KNFda tasvirlanadi. Bunday holda funksional sxemalarda elementlar soni minimal bo‘lib, sxemalarni ko‘rish oson kechadi.

Mantiqiy funksiyalarni HAMDA-YOKI-EMAS bazisida yozish quyidagicha bajariladi:

HAMDA-YOKI-EMAS bazisida berilgan funksiyaning invers qiymati minimallashtiriladi, so‘ngra berilgan funksiyaning olingan natijasiga inversiya qo‘yiladi va (2.16) de Morgan qonunlari yordamida HAMDA-YOKI-EMAS bazisiga o‘tiladi.

2.7-misol. Mantiqiy funksiyaning minimal DNFni HAMDA-YOKI-EMAS bazisida yozing:

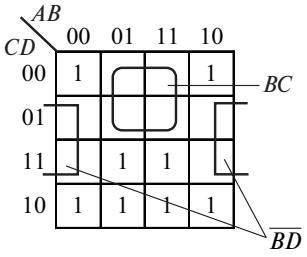
$$X=(\bar{A}\bar{B}\bar{C}\bar{D})\vee(\bar{A}\bar{B}CD)\vee(\bar{A}BC\bar{D})\vee(\bar{A}BCD)\vee$$
$$\vee(ABC\bar{D})\vee(ABCD)\vee(\bar{A}\bar{B}\bar{C}\bar{D})\vee(A\bar{B}C\bar{D}).$$

Yechish: 1. Berilgan funksiyani Karno kartalari yordamida minimallashtiramiz (2.12-rasm). Bo‘sh kataklarga mos kelgan mintermlarni guruhlab yozamiz:

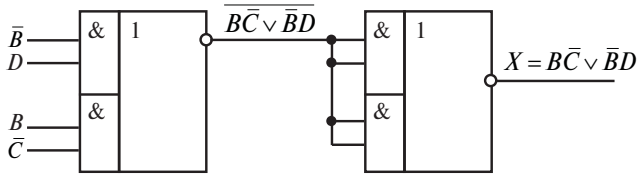
$$X_{\text{DNF}}=\overline{(B\bar{C})\vee(\bar{B}D)}.$$

2. Shu ifodadan inversiya olib, HAMDA-YOKI-EMAS bazisiga o‘tamiz: $X_{\text{DNF}}=(B\bar{C})\vee(\bar{B}D)$.

Bu mantiqiy funksiyani amalga oshiradigan funksional sxema 2.13-rasmda ko‘rsatilgan.



2.12-rasm. Karno kartasi yordamida $X=(\bar{A}\bar{B}\bar{C}\bar{D})\vee(\bar{A}\bar{B}CD)\vee(\bar{A}BC\bar{D})\vee$ $\vee(\bar{A}BCD)\vee(ABC\bar{D})\vee(ABCD)\vee(\bar{A}\bar{B}\bar{C}\bar{D})\vee(A\bar{B}C\bar{D})$ funksiyani minimallashtirish.



2.13-rasm. $X_{\text{DNF}}=(B\bar{C})\vee(\bar{B}D)$ mantiqiy funksiyaning funksional sxemasi.

Nazorat savollari va topshiriqlari

1. Mantiqiy funksiyalarni HAMDA-EMAS va YOKI-EMAS universal bazislarida qaysi tartibda yozish kerak?
2. DNFda HAMDA-YOKI-EMAS bazisida

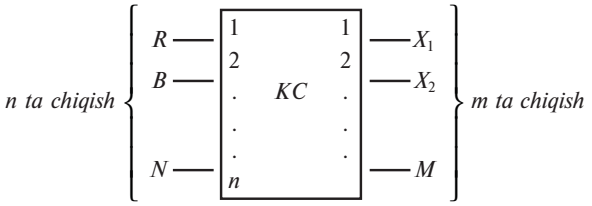
$$X=(\bar{A}\bar{B}\bar{C}D)\vee(AB\bar{C}D)\vee(A\bar{B}CD)\vee(ABCD)$$

mantiqiy funksiyani yozing.

2.8. KOMBINATSION SXEMALARNING ANALIZI VA SINTEZI

Kombinatsion sxema (KS) (2.14-rasm) to‘liq berilgan bo‘lishi uchun uning faoliyati quyidagi mantiqiy funksiyalar tizimi bilan ifodalangan bo‘lishi kerak:

$$\left.\begin{array}{l} X_1=f_1(A,B,C,...,N);\\ X_2=f_2(A,B,C,...,N);\\ \cdots\cdots\cdots\cdots\cdots\cdots\\ X_n=f_n(A,B,C,...,N) \end{array}\right\}$$



2.14-rasm. Kombinatsion sxemaning shartli grafik ko‘rinishi.

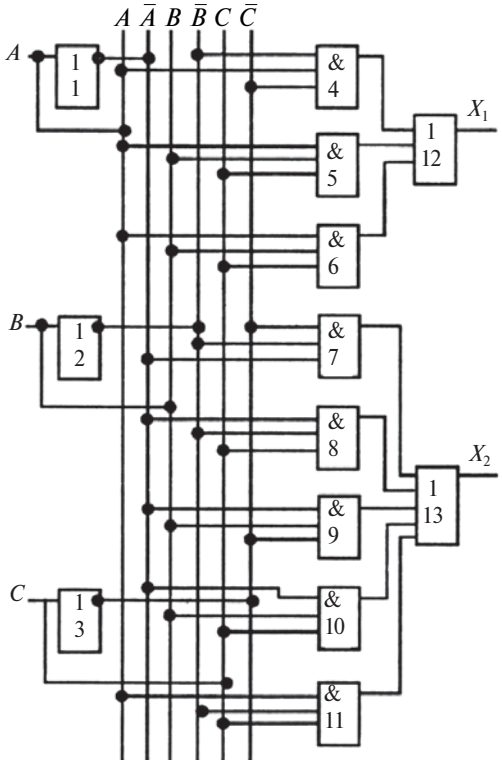
Kombinatsion sxemalarning (KS) analizi. Kombinatsion sxemalarning analizi quyidagi tartibda amalga oshiriladi:

1. Berilgan kombinatsion sxemadagi har bir element ishini ketma-ket mantiqiy funksiyalar orqali yozib, uning ishlash qonunini yorituvchi mantiqiy funksiyalar olinadi.

2. Ortiqcha elementlarni yo‘qotish uchun olingan mantiqiy funksiyalar analiz qilinadi.

2.8-misol. Rasmda keltirilgan kombinatsion sxemaning mantiqiy strukturasini analiz qiling (2.15-rasm).

Yechish: 1) Mantiqiy funksiya bilan berilgan KS har bir mantiqiy elementining ketma-ket ishlatilishini yozamiz:



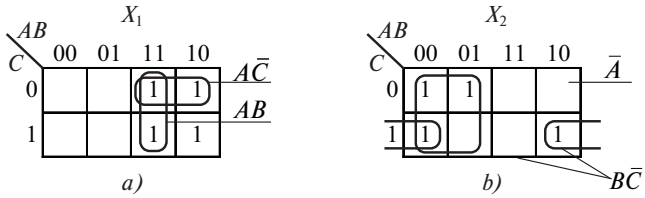
2.15-rasm. Kombinatsion sxemaning mantiqiy strukturasi.

$$X_1 = A\bar{B}\bar{C} \vee AB\bar{C} \vee ABC;$$
$$X_2 = \bar{A}\bar{B}\bar{C} \vee \bar{A}\bar{B}C \vee \bar{A}B\bar{C} \vee A\bar{B}C.$$

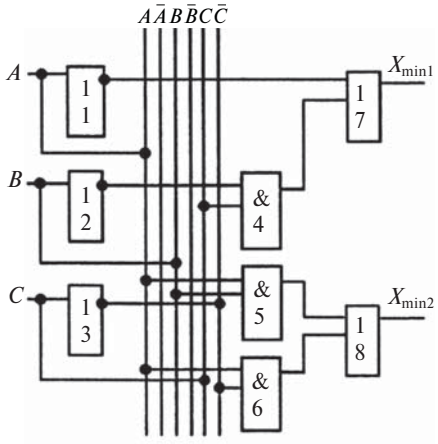
2) Yuqoridagi natijalardan foydalanib, Karno kartalarini tuzamiz (2.16-rasm) va KSga tegishli mantiqiy funksiyalarning minimal DNFlarini yozamiz:

$$X_{\min 1} = AB \vee A\bar{C};$$
$$X_{\min 2} = A \vee \bar{B}C.$$

2.17-rasmda mantiqiy funksiyalar minimal formalarining mantiqiy sxemasi keltirilgan. Hosil bo‘lgan KSda, berilgan KSga nisbatan HAMDA mantiqiy elementi hamda kirish yo‘li kam ishlatilgan.



2.16-rasm. $X_1 = A\bar{B}\bar{C} \vee ABC$; $X_2 = \bar{A}\bar{B}\bar{C} \vee \bar{A}\bar{B}C \vee \bar{A}B\bar{C} \vee A\bar{B}C$ funksiyalar uchun Karno kartalari.



2.17-rasm. $X_1 = A\bar{B}\bar{C} \vee ABC$; $X_2 = \bar{A}\bar{B}\bar{C} \vee \bar{A}\bar{B}C \vee \bar{A}BC \vee A\bar{B}C$ minimal formalarning mantiqiy sxemasi.

Kombinatsion sxemalarning sintezi. *Sintez – berilgan funksional qonunni amalga oshiruvchi KSni loyihalashdir.* Sintez bosqichlarini quyidagi misolda ko‘rib chiqamiz.

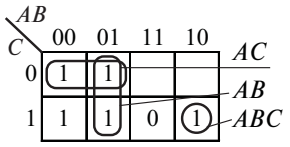
2.9-misol. HAMDA-EMAS bazisida kombinatsion sxemani tuzing. KSning funksional qonuni jadvalda berilgan (14-jadval).

Yechish: 1. 14-jadvaldan foydalanib HAMDA, YOKI, EMAS bazisida kombinatsion sxemaning mantiqiy funksiyasini yozamiz:

$$X_{\text{MDNF}} = \bar{A}\bar{B}\bar{C} \vee \bar{A}B\bar{C} \vee \bar{A}B\bar{C} \vee A\bar{B}C .$$

2. Karno kartasi yordamida olingan mantiqiy funksiyani minimallashtiramiz (2.18-rasm):

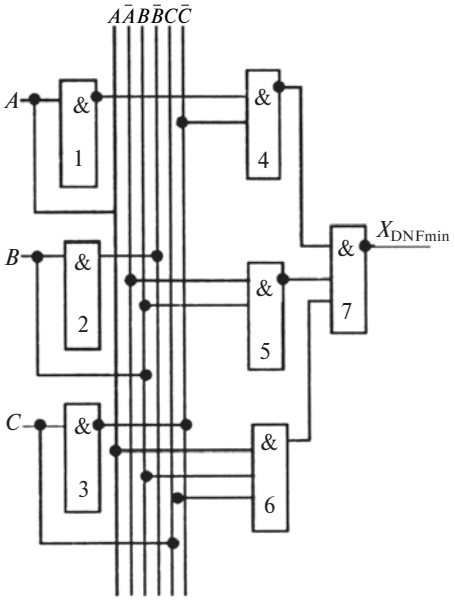
$$X_{\text{DNFmin}} = \bar{A}\bar{C} \vee \bar{A}B \vee A\bar{B}C .$$



2.18-rasm. Karno kartasi yordamida $X_{\text{MDNF}} = \bar{A}\bar{B}\bar{C} \vee \bar{A}B\bar{C} \vee \bar{A}BC \vee A\bar{B}C$ funksiyani minimallashtirish.

14-jadval

A	*B*	*C*	*X*
0	0	0	1
0	0	1	0
0	1	0	1
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	0
1	1	1	0



2.19-rasm. $X_{\text{DNFmin}} = \overline{\overline{\bar{A}\bar{C}} \vee \overline{\bar{A}B} \vee \overline{A\bar{B}C}}$ mantiqiy funksiyaning funksional sxemasi.

3. X_{DNFmin} ni HAMDA-EMAS bazisida yozamiz:

$$X_{\text{DNFmin}} = \overline{\overline{\overline{A}\overline{C} \vee \overline{A}B \vee A\overline{B}C}} = \overline{(\overline{A}\overline{C}) \vee (\overline{A}B) \vee (A\overline{B}C)}.$$

4. Sheffer elementlaridan iborat kombinatsion sxemani tuzamiz (2.19-rasm).

Kombinatsion sxemaning optimal variantini tanlashda real mantiqiy elementlar kartasiga yuklanuvchi tarmoqlanish koeffitsiyenti, mantiqiy elementning kirishlar soni va mantiqiy elementdagi signalni tarqatishning oxirgi vaqti kabi cheklanishlarni hisobga olish zarurdir.

Nazorat savollari va topshiriqlari

1. $X = A\overline{B}\overline{C} \vee ABC \vee AB\overline{C} \vee ABC$ funksiyani HAMDA-EMAS DNF normal bazisida yozing va Sheffer elementlari yordamida kombinatsion sxemani sintezlang.
2. $X = \overline{A}\overline{B}\overline{C}\overline{D} \vee AB\overline{C}D \vee A\overline{B}CD$ funksiyani HAMDA-YOKI, EMAS DNF normal bazisida yozing va HAMDA-YOKI-EMAS elementlari yordamida kombinatsion sxemani sintezlang.

3-bob. KOMBINATSIYALASHGAN QURILMALARNING SXEMOTEXNIKASI

3.1. DESHIFRATORLAR VA SHIFRATORLAR

Deshifrator (DC – decoder) – kirish yo‘lidagi signallarni faqat chiqish yo‘lining bittasiga chiqarib beruvchi raqamli qurilmalarning (EHMlarning) uzelidir. Boshqacha qilib aytganda, deshifrator ikkilik kodni unitar kodga o‘zgartirib beruvchi qurilmadir. EHMlarda deshifratorlar ikkilik kodlarni o‘nlik kodlarga o‘zgartirib berish uchun xizmat qiladi.

Deshifratorlardan kichik hajmdagi doimiy xotiralarni loyihalashda foydalanish mumkin. Shuningdek, deshifrator va multipleksorni hamkorlikda qo‘llash natijasida ikkita ko‘p razryadli ikkilik kodning komporatorini yoki matritsali summatorlarni qurish mumkin. Deshifratorlar matritsali kommutatorlarni qurishda ham keng qo‘llaniladi.

To‘la deshifratorda chiqish yo‘llari soni $m = 2^n$ bo‘lib, unda n – kirish yo‘llari sonini ifodalaydi. To‘la bo‘lmagan deshifratorda esa $m < 2^n$.

To‘la deshifrator deb, n kirish yo‘liga va 2^n chiqish yo‘liga ega bo‘lgan hamda 2^n mantiqiy funksiyalarni amalga oshiruvchi kombinatsion sxemaga aytiladi:

$$y_1 = \overline{x}_n \cdot \overline{x}_{n-1} \cdot \ldots \cdot \overline{x}_2 \cdot \overline{x}_1 ,$$

$$y_2 = \overline{x}_n \cdot \overline{x}_{n-1} \cdot \ldots \cdot \overline{x}_2 \cdot x_1 ,$$

$$y_3 = \overline{x}_n \cdot \overline{x}_{n-1} \cdot \ldots \cdot x_2 \cdot x_1 ,$$

$$\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots,$$

$$y_m = x_n \cdot x_{n-1} \cdot \ldots \cdot x_2 \cdot x_1 ,$$

bu yerda: $m = 2^n$.

Har bir y_i mantiqiy funksiya birning konstituentasi bo‘lib hisoblanadi.

So‘zni deshifratsiyalashni tashkil etilishiga qarab deshifratorlar to‘g‘ri burchakli (matritsali), piramidal va pog‘onali (kaskadli) turlarga bo‘linadi.

To‘rtta kirish yo‘liga ega bo‘lgan to‘la deshifratorning rostlik jadvalini (o‘tish jadvalini) tuzamiz (15-jadval).

15-jadval

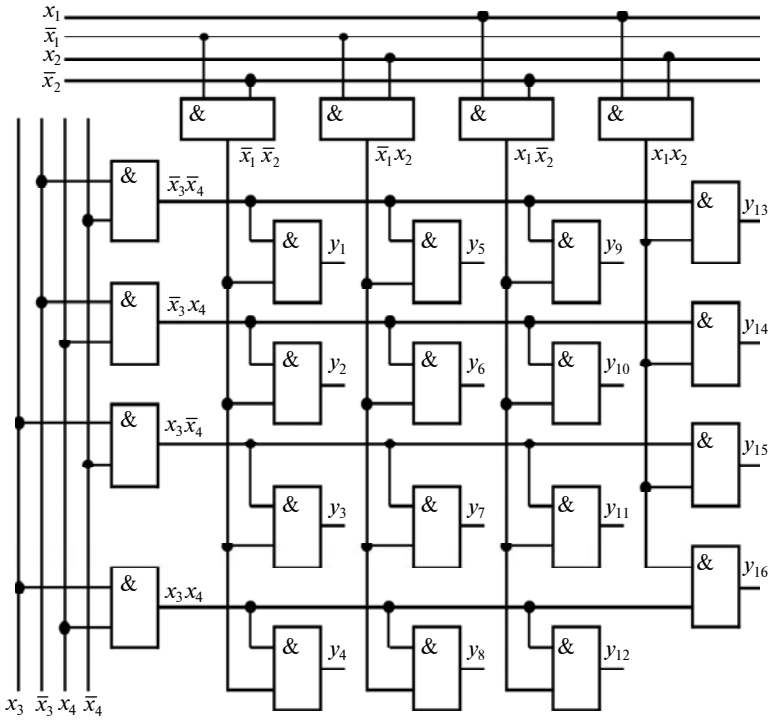
x_1	x_2	x_3	x_4	y_1	y_2	y_3	y_4	y_5	y_6	y_7	y_8	y_9	y_{10}	y_{11}	y_{12}	y_{13}	y_{14}	y_{15}	y_{16}
0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	1	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	1	1	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0
0	1	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0
0	1	0	1	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0
0	1	1	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0
0	1	1	1	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0
1	0	0	1	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0
1	0	1	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0
1	0	1	1	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0
1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0
1	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0
1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0
1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1

Chiqish signallari uchun mantiqiy ifodalar quyidagi ko‘rinishga ega bo‘ladi:

$y_1 = \overline{x}_1 \cdot \overline{x}_2 \cdot \overline{x}_3 \cdot \overline{x}_4$	$y_6 = \overline{x}_1 \cdot x_2 \cdot \overline{x}_3 \cdot x_4$	$y_{11} = x_1 \cdot \overline{x}_2 \cdot x_3 \cdot \overline{x}_4$
$y_2 = \overline{x}_1 \cdot \overline{x}_2 \cdot \overline{x}_3 \cdot x_4$	$y_7 = \overline{x}_1 \cdot x_2 \cdot x_3 \cdot \overline{x}_4$	$y_{12} = x_1 \cdot \overline{x}_2 \cdot x_3 \cdot x_4$
$y_3 = \overline{x}_1 \cdot \overline{x}_2 \cdot x_3 \cdot \overline{x}_4$	$y_8 = \overline{x}_1 \cdot x_2 \cdot x_3 \cdot x_4$	$y_{13} = x_1 \cdot x_2 \cdot \overline{x}_3 \cdot \overline{x}_4$
$y_4 = \overline{x}_1 \cdot \overline{x}_2 \cdot x_3 \cdot x_4$	$y_9 = x_1 \cdot \overline{x}_2 \cdot \overline{x}_3 \cdot \overline{x}_4$	$y_{14} = x_1 \cdot x_2 \cdot \overline{x}_3 \cdot x_4$
$y_5 = \overline{x}_1 \cdot x_2 \cdot \overline{x}_3 \cdot \overline{x}_4$	$y_{10} = x_1 \cdot \overline{x}_2 \cdot \overline{x}_3 \cdot x_4$	

Bu ifodalar to‘g‘ri burchakli (matritsali) deshifratorda sxematik tarzda 3.1-rasmdagi kabi amalga oshiriladi.

Chiqish yo‘li $m < 2^n$ bo‘lgan n-razryadli ikkilik kodning deshifratori to‘la bo‘lmagan deshifrator deyiladi. Bunday deshifratorning sxemasini qurishda ahamiyatsiz bo‘lgan kombinatsiyalar sxemani soddalashtirishda qo‘llanilishi mumkin.



3.1-rasm. Matritsali deshifratorning funksional sxemasi.

Masalan, quyidagi $x = x_4 \cdot x_3 \cdot x_2 \cdot x_1$ to‘rt razryadli ikkilik kodning deshifratori oltita kombinatsiyani ajratishda quyidagicha foydalaniladi:

16-jadval

0000	$y_1 = \bar{x}_1 \cdot \bar{x}_2 \cdot \bar{x}_3 \cdot \bar{x}_4$
0101	$y_2 = x_1 \cdot \bar{x}_2 \cdot x_3 \cdot \bar{x}_4$
1001	$y_3 = x_1 \cdot \bar{x}_2 \cdot \bar{x}_3 \cdot x_4$
1010	$y_4 = \bar{x}_1 \cdot x_2 \cdot \bar{x}_3 \cdot x_4$
1100	$y_5 = \bar{x}_1 \cdot \bar{x}_2 \cdot x_3 \cdot x_4$
1111	$y_6 = x_1 \cdot x_2 \cdot x_3 \cdot x_4$

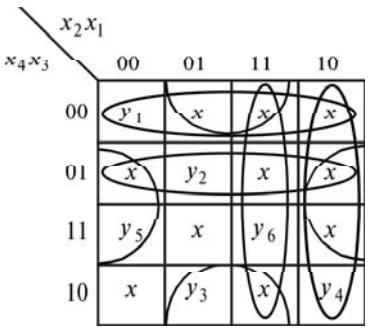
Qolgan o‘nta kombinatsiya ahamiyatsiz hisoblanadi.

Barcha kombinatsiyalarni Karno kartasiga joylashtiramiz (3.2-rasm).

Karno kartasi tahlili asosida quyidagi ifodalarga ega bo‘lamiz:

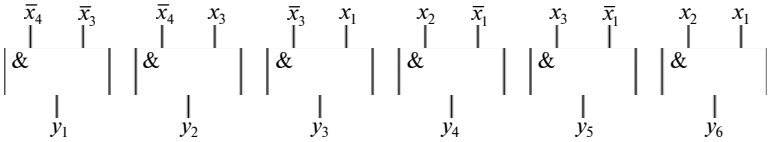
$$y_1 = \bar{x}_4\bar{x}_3 \quad y_3 = \bar{x}_3x_1 \quad y_5 = x_3\bar{x}_1$$

$$y_2 = \bar{x}_4x_3 \quad y_4 = x_2\bar{x}_1 \quad y_6 = x_2x_1$$



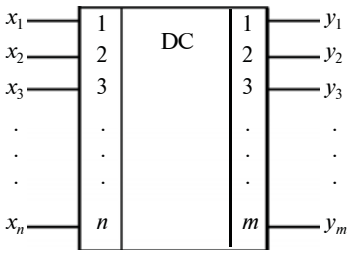
3.2-rasm. Karno kartasi.

Unda deshifratorning sxemasi 3.3-rasmdagi kabi ko‘rinishga ega bo‘ladi.



3.3-rasm. To‘la bo‘lmagan deshifratorning sxemasi.

Deshifratorning shartli belgilanishi 3.4-rasmdagi kabi ko‘rinishga ega.



3.4-rasm. Deshifratorning shartli belgilanishi.

Shifrator (CD – coder) – kirish yo‘lidagi birlik signalni *n* razryadli ikkilik kodga o‘zgartiradigan raqamli qurilmalarning uzelidir.

Boshqacha qilib aytganda, shifrator o‘nlik kodni ikkilik kodga aylantirib berish uchun xizmat qiladigan operatsion elementdir. Shuning uchun shifratorlar raqamli texnikaning va EHMlarning axborotlarni kiritish qurilmalarida o‘nlik kodlarni ikkilik kodlarga o‘zgartirishda qo‘llaniladi.

Shifratorning kirish va chiqish yo‘llari soni $k = 2n$ munosabat bilan belgilanadi va u deshifratorga nisbatan teskari funksiyani bajaradi.

Masalan, $k = 10$ kirish yo‘liga ega bo‘lgan va chiqish kodi tabiiy darajadagi ikkilik kodga ega bo‘lsin. U holda $n = 4$ bo‘lib, rostlik jadvali 17-jadvaldagi ko‘rinishga ega bo‘ladi.

17-jadval

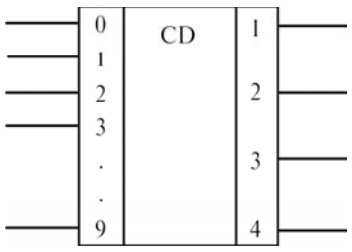
x_0	x_1	x_2	x_3	x_4	x_5	x_6	x_7	x_8	x_9	y_4	y_3	y_2	y_1
1	0	0	0	0	0	0	0	0	0	0	0	0	0
0	1	0	0	0	0	0	0	0	0	0	0	0	1
0	0	1	0	0	0	0	0	0	0	0	0	1	0
0	0	0	1	0	0	0	0	0	0	0	0	1	1
0	0	0	0	1	0	0	0	0	0	0	1	0	0
0	0	0	0	0	1	0	0	0	0	0	1	0	1
0	0	0	0	0	0	1	0	0	0	0	1	1	0
0	0	0	0	0	0	0	1	0	0	0	1	1	1
0	0	0	0	0	0	0	0	1	0	1	0	0	0
0	0	0	0	0	0	0	0	0	1	1	0	0	1

Rostlik jadvalidan quyidagilarni aniqlaymiz:

$$y_1 = x_1 + x_3 + x_5 + x_7 + x_9,$$
$$y_2 = x_2 + x_3 + x_6 + x_7,$$
$$y_3 = x_4 + x_5 + x_6 + x_7,$$
$$y_4 = x_8 + x_9.$$

Mantiq algebrasi qonuniyatlari asosida ifodalarni quyidagi ko‘rinishlarga o‘zgartiramiz:

$$y_1 = \overline{\overline{x}_1 \cdot \overline{x}_3 \cdot \overline{x}_5 \cdot \overline{x}_7 \cdot \overline{x}_9},$$
$$y_2 = \overline{\overline{x}_2 \cdot \overline{x}_3 \cdot \overline{x}_6 \cdot \overline{x}_7},$$
$$y_3 = \overline{\overline{x}_4 \cdot \overline{x}_5 \cdot \overline{x}_6 \cdot \overline{x}_7},$$
$$y_4 = \overline{\overline{x}_8 \cdot \overline{x}_9}.$$

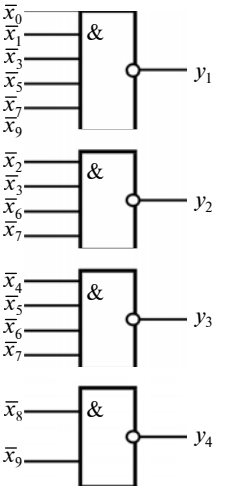


3.5-rasm. Shifratorning shartli belgilanishi.

Shifratorning shartli belgilanishi 3.5-rasmda keltirilgan.

HAMDA-EMAS elementi asosida shifratorning sxemasi 3.6-rasmda keltirilgan.

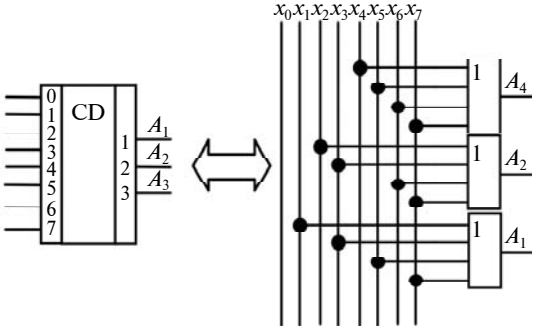
3.6-rasm. HAMDA-EMAS elementi asosidagi shifratorning sxemasi.



Kirish yo‘li ko‘p bo‘lgan holda, kirish bo‘yicha birlashtirish koeffitsiyentini kamaytirish uchun kaskadli shifratorlarni qurish maqsadga muvofiq bo‘ladi.

Shifratorni HAMDA mantiqiy elementlar bazasida ham qurish mumkin. Bunday shifratorning rostlik jadvali, mantiqiy funksiyalari, grafik belgilanishi hamda funksional sxemasi 3.7-rasmda keltirilgan.

kirish	A_4	A_2	A_1
0	0	0	0
1	0	0	1
2	0	1	0
3	0	1	1
4	1	0	0
5	1	0	1
6	1	1	0
7	1	1	1



$$A_4 = X_4 \vee X_5 \vee X_6 \vee X_7$$
$$A_2 = X_2 \vee X_3 \vee X_6 \vee X_7$$
$$A_1 = X_1 \vee X_3 \vee X_5 \vee X_7$$

3.7-rasm. HAMDA mantiqiy elementi bazisida qurilgan shifrator.

Nazorat savollari va topshiriqlari

1. Matritsali deshifrator to‘la deshifratordan nima bilan farq qiladi?
2. Shifratorlar HAMDA mantiqiy elementlar bazasida qanday quriladi?
3. To‘la bo‘lmagan deshifratorning ishlash prinsipini tushuntiring.
4. Shifratorning rostlik jadvali qanday aniqlanadi?

3.2. MULTIPLEKSORLAR VA DEMULTIPLEKSORLAR

Multipleksor (MUX − multiplexor) − raqamli qurilmalarning, shu jumladan, EHMlarning uzeli bo‘lib, parallel raqamli kodlarni ketma-ket kodlarga o‘zgartirish uchun xizmat qiladi.

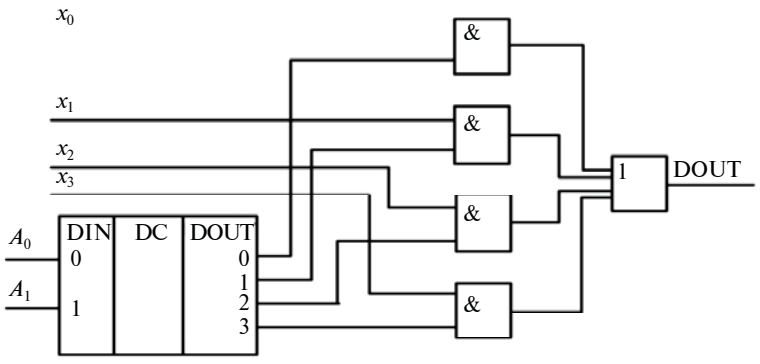
Multipleksorning kirish yo‘llari ikkiga bo‘linadi: axborot kirish yo‘li va boshqaruvchi (manzilli) kirish yo‘li.

18-jadvalda multipleksorning rostlik jadvali keltirilgan. Bu jadvalda $m = 4$ holat uchun oddiy multipleksorning faoliyati tahlil etilgan. Bu yerda: $x_0 ... x_3$ − axborotlarning mustaqil kirish yo‘llari; A_1 va A_0 − boshqaruvchi kodlar; Y − axborotning chiqishi.

18-jadval

Axborotning kirish yo‘li	Boshqaruvchi kodlar		Chiqish
	A_1	A_0	Y (DOUT)
x_0	0	0	x_0
x_1	0	1	x_1
x_2	1	0	x_2
x_3	1	1	x_3

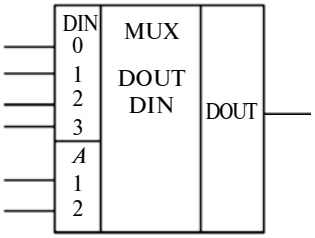
Multipleksorning rostlik jadvali asosida uning sxemasini quramiz (3.8-rasm).



3.8-rasm. Oddiy multipleksorning sxemasi.

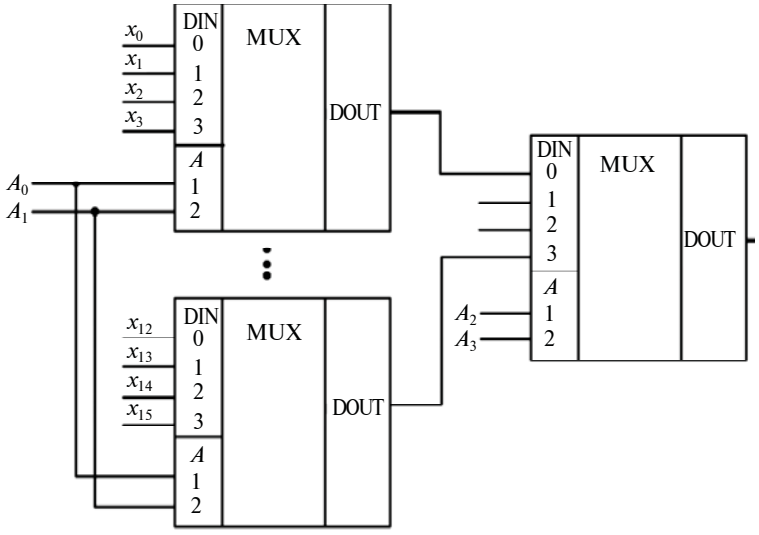
Ushbu multipleksor quyidagicha ishlaydi: x_0, x_1, x_2 va x_3 parallel kodlar to‘rt takt mobaynida A_1 va A_0 boshqaruvchi kodlarning qiymatiga qarab ketma-ket chiqish yo‘liga uzatiladi. Multipleksor tarkibidagi DC manzillar deshifratori kirish signallarini (ikkilik kodlarini) chiqish yo‘liga uzatishni boshqaradi. Kirish signallari soni oshganda manzillar deshifratorining boshqaruvchi kodlari soni ham oshishi kerak.

Multipleksorlardan universal mantiqiy modullar rejimida ham foydalanish mumkin. Multipleksorlashtirish funksiyasining shartli grafik belgilanishida MUX (multiplexor so‘zidan olingan) belgisi ishlatiladi (3.9-rasm).

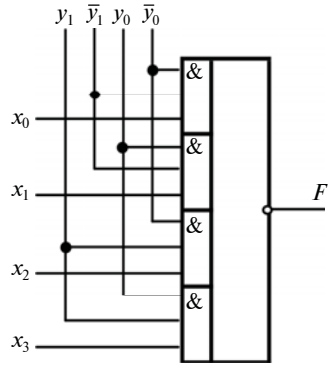


3.9-rasm. Oddiy multipleksorning shartli belgilanishi.

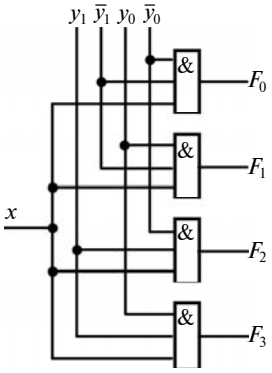
Axborotlarning o‘tish manbalari sonini (kirish signallari soni) oshirish uchun multipleksorlarda ko‘p pog‘onali ulanishni tashkil etish orqali amalga oshirish mumkin. Chunki, birinchi darajali multipleksorlarning chiqish yo‘li ikkinchi darajali multipleksorlarning kirish yo‘liga ulangan taqdirda multipleksorli «daraxt» hosil bo‘ladi (3.10-rasm).



3.10-rasm. Multipleksorli «daraxt» sxemasi.



3.11-rasm. Multipleksorning HAMDA-YOKI-EMAS elementlar bazasida qurilgan sxemasi.

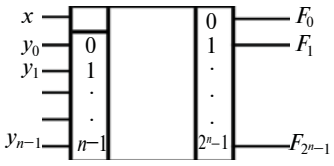


3.12-rasm. HAMDA mantiqiy elementlar asosida qurilgan demultipleksorning sxemasi.

Multipleksorlarning «4→1» sxemada HAMDA-YOKI-EMAS elementlar bazasida qurilgan sxemasi 3.11-rasmdagi ko‘rinishga ega.

Bu sxemada $x_0 \cdot x_1 \cdot x_2 \cdot x_3$ axborotning kirish yo‘llarini ifodalaydi, y_1,y_0 – boshqaruvchi kodlar.

Demultipleksor. Raqamli qurilmalarda multipleksorlarning ishiga qarama-qarshi operatsiyalarni bajaruvchi demultipleksorlar ham keng qo‘llaniladi. Demultipleksorlar ketma-ket kodlarni parallel kodlarga o‘tkazish uchun xizmat qiladi.



3.13-rasm. Demultipleksorning shartli belgilanishi.

HAMDA mantiqiy elemenlar bazasida «1→4» sxema asosida qurilgan demultipleksorning sxemasi 3.12-rasmda keltirilgan.

Demultipleksor bitta axborot kirish yo‘liga, n boshqaruvchi (manzilli) kirish yo‘liga va 2^n chiqish yo‘liga ega.

Demultipleksorning shartli belgilanishi 3.13-rasmda keltirilgan.

Multipleksorlar va demultipleksorlarga nisbatan «ma’lumotlar selektori» atamasi ham keng qo‘llaniladi.

Nazorat savollari va topshiriqlari

1. Multipleksor iborasiga ta’rif bering.
2. Multipleksorni qaysi elementlar asosida qurish mumkin?

3. Demultipleksorning ishlash prinsipini tushuntiring.
4. Multipleksorlar va demultipleksor EHMlarning qaysi qurilmalarida ishlatiladi?
5. Multipleksor va demultipleksorning umumiy va xususiy tomonlarini ko‘rsating.
6. HAMDA mantiqiy elementlar asosida qurilgan demultipleksorning ishlash prinsipini tushuntiring.

3.3. SUMMATORLAR

Ikki son xonalarini jamlash amalini bajaruvchi EHM uzeli *summator* deb ataladi.

Summatorlarni quyidagi belgilari bo‘yicha tasniflash mumkin:

1) bir xonali sonlarni jamlash usuli bo‘yicha kombinatsion va to‘plovchi summatorlar;

2) bir xonali sonlarni jamlash sxemasidagi kirish yo‘llari soni bo‘yicha: ikki kirish yo‘lli bir xonali (yarim summatorlar) va uch kirish yo‘lli bir xonali summatorlar;

3) ko‘p xonali sonlarni jamlash usuli bo‘yicha: ketma-ket va parallel summatorlar;

4) sanoq tizimining asosi va qabul qilingan kodlash usuli bo‘yicha: ikkilik, uchlik, o‘nlik va ikkilik-o‘nlik summatorlar;

5) ko‘chirish zanjirini tashkil qilish usuli bo‘yicha: ketma-ket, bir boshdan, bir vaqtda, guruhli, shartli ko‘chirishli va ko‘chirish qiymati signalini xotirada saqlovchi summatorlar.

Biz yuqorida sanab o‘tgan summatorlarning har biri ma’lum ma’noda o‘zining yutuq va kamchiliklariga ega.

Summatorlarda ikkilik sonlarni qo‘shish deganda ikkita x (x_1, x_2, ..., x_n) va y (y_1, y_2, ..., y_n) qo‘shiluvchilarning o‘zaro qo‘shilishi natijasida s (s_1, s_2, ..., s_n) yig‘indining hosil bo‘lishi tushuniladi.

Qo‘shish jarayonida sonlarning xonadagi qiymati quyidagi qonuniyat asosida hosil bo‘ladi:

$$\left.\begin{aligned} S_i &= x_i + y_i + P_{i-1}; \\ P_i &= 0 \end{aligned}\right\} \quad (x_i + y_i + P_{i-1}) \text{ bo‘lganda;}$$

$$\left.\begin{aligned} S_i &= x_i + y_i + P_{i-1} - q; \\ P_i &= 1 \end{aligned}\right\} \quad (x_i + y_i + P_{i-1}) \geq q \text{ bo‘lganda,}$$

bu yerda: S_i – i-razryadda (xonada) hosil bo‘lgan yig‘indi; P_{i-1} – oldingi kichik razryaddan keyin kelgan ikkilik kod; P_i – keyingi katta razryadga o‘tadigan ikkilik kod; q – sanoq tizimining asosi.

Ketma-ket summatorlar. Ketma-ket summatorlar ikkita ikkilik kodni xonama-xona qo‘yish uchun xizmat qiladi. Shuning uchun ular *bir razryadli (bir xonali) summatorlar* deyiladi. Bir razryadli ketma-ket summatorning o‘tish jadvalini tuzamiz (19-jadval).

19-jadval

Kirish			**Chiqish**	
Qo‘shiluvchilar		**Oldingi kichik razryaddan o‘tish P_{i-1}**	**Yig‘indi**	**Keyingi katta razryadga o‘tish P_{i+1}**
X_1	X_2	X_3	S	P_{i+1}
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

Jadvaldagi S_i va P_{i+1} ifodalar uchun dizyunktiv normal forma (DNF) quyidagicha ifodalanadi:

$$S_i = \overline{X}_1 X_2 \overline{X}_3 \vee X_1 \overline{X}_2 \overline{X}_3 \vee \overline{X}_1 \overline{X}_2 X_3 \vee X_1 X_2 X_3,$$

$$P_{i+1} = X_1 X_2 \overline{X}_3 \vee \overline{X}_1 X_2 X_3 \vee X_1 \overline{X}_2 X_3 \vee X_1 X_2 X_3.$$

Ushbu kanonik formalar bo‘yicha ketma-ket summatorning sxemasini HAMDA, shuningdek, YOKI mantiqiy elementlaridan foydalanib ko‘rish mumkin (3.14-rasm).

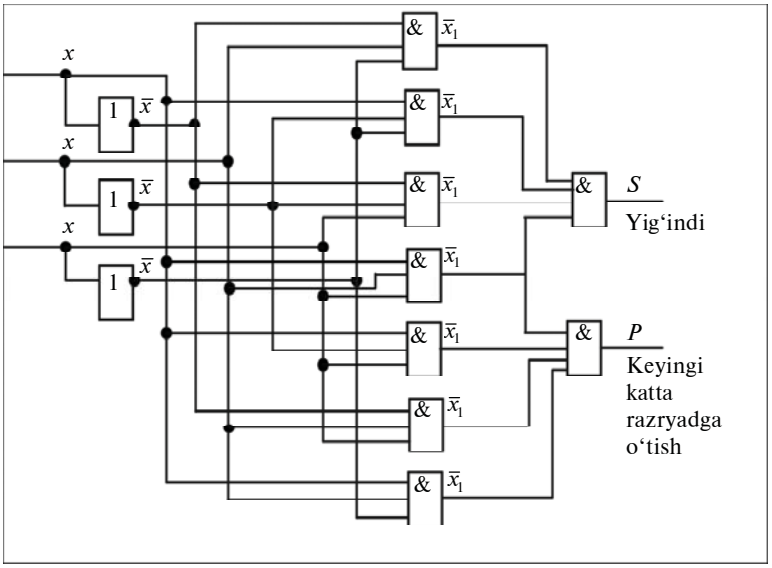
Sxemaning kirish yo‘llarida x_1, x_2, x_3 signallar bilan bir qatorda ularning invers qiymatlari ham ishlatiladi.

Ketma-ket summatorning funksional sxemasi to‘la bo‘lishi uchun chiqish yo‘lidagi P_{i+1} o‘tish signalini x_3 kirish yo‘liga bir takt vaqt mobaynida ushlagan holda ulash talab etiladi. Unga ko‘ra bir razryadli ketma-ket summatorning sxemasini keltiramiz (3.15-rasm).

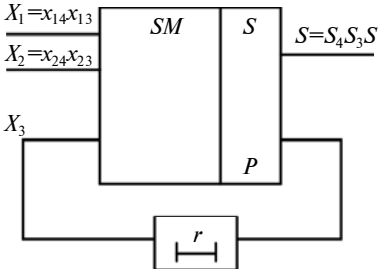
Summator tomonidan ikkilik kodlarni qo‘shishga sarflangan vaqt quyidagicha aniqlanadi:

$$T_c \approx n - \Delta t,$$

bu yerda: n – razryadlar soni; Δt – har bir razryadni qo‘shish uchun ketgan vaqt.



3.14-rasm. Ketma-ket summatorlarning funksional sxemasi.



3.15-rasm. Ketma-ket summatorning prinsipial sxemasi.

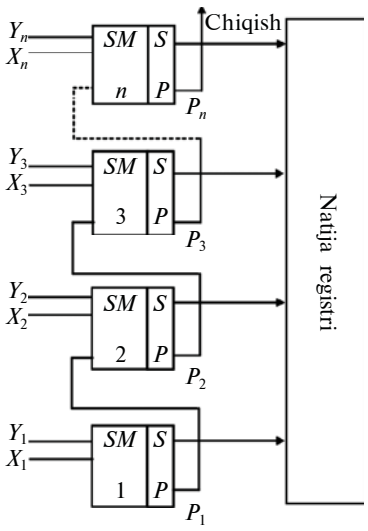
Formuladan ko'rinib turibdiki, bir razryadli ketma-ket summatorning asosiy kamchiligi uning tezligining past ko'rsatkichidir. Uning yutug'i esa elementlar sonining kamligi va tejamkorligidir.

Ko'p razryadli parallel summatorlar. EHMlarning tezligini oshirish va parallel ikkilik kodlarning qayta ishlashi uchun ko'p razryadli parallel summatorlar qo'llaniladi.

Parallel summatorlar soni qo'shiluvchilar xonalarining soniga teng bo'lgan bir xonali summatorlar asosida quriladi va unda qo'shiluvchilar kodining barcha xonalari bir vaqtda ishlanadi.

Ko'p razryadli parallel summatorning sxemasini keltiramiz (3.16-rasm).

Bu sxema uch kirish yo'lli va *n* bir xonali kombinatsion summatorlardan tashkil topgan.



3.16-rasm. Ko‘p razryadli parallel summatorning sxemasi.

Summatorning kirish yo‘llariga qo‘shiluvchilarning mos xonalari (x_n va y_n), oldingi (kichik) xonadan ko‘chirish qiymati signali (P_i) beriladi.

Har qaysi bir xonali summator chiqish yo‘llarida xona yig‘indisi raqami kodining signali hamda keyingi (katta) xonaga ko‘chirish qiymatining signali shakllanadi.

Sxemadan ko‘rinib turibdiki, biror xonada paydo bo‘lgan ko‘chirish qiymatining signali yuqori xonalarga summatorlar orqali ketma-ket tarqaladi.

Agar birlardan iborat bo‘lgan son bilan faqat birinchi xonasi birga teng bo‘lgan son qo‘shilsa, birinchi xonada paydo bo‘lgan ko‘chirish qiymati signalining tarqalish zanjiri barcha summatorlarni o‘z ichiga oladi.

Har qaysisi bir xonali summatorlarda ko‘chirish qiymatining signali kirish signallari (x_i, y_i, P_i) berilishi paytida ma’lum vaqtga kechikishi bilan shakllanganligi sababli bunday summatorning tezligi quyidagicha aniqlanadi:

$$T = t_{c-i} + (n - 1)t_{p.t.},$$

bu yerda: t_{c-i} – razryadda qo‘shish uchun ketgan vaqt; $t_{p.t.}$ – ko‘chirishning n razryadli summatordan ketma-ket o‘tishiga ketgan vaqt.

Nazorat savollari va topshiriqlari

1. Summatorlarning qanday turlarini bilasiz?
2. Parallel va ketma-ket summatorlarning umumiy va xususiy tomonlarini ko‘rsating.
3. Summatorlarni qaysi elementlar asosida qurish mumkin?
4. Ketma-ket summator qanday kamchiliklarga ega?

4- bob. KETMA-KET RAQAMLI QURILMALARNING SXEMOTEXNIKASI

4.1. TRIGGERLAR

Sonli avtomatlarda funksiya qiymati faqat o‘zgaruvchilarning joriy taktdagi qiymatigagina bog‘liq bo‘lmay, balki oldingi taktdagi qiymatlariga ham bog‘liqdir. Demak, sonli avtomatlarda bir va undan ortiq taktdagi signallarni o‘zida saqlashi talab etiladi. EHM larda bu vazifalarni, asosan, triggerlar bajaradi.

Trigger – ikkita teng kuchli, alternativ turg‘un holatga ega bo‘lgan (0 yoki 1) va axborotni yozish, saqlash va uzatish uchun xizmat qiladigan qurilmadir.

Boshlang‘ich signallar ta’sirida trigger bir turg‘un holatdan ikkinchisiga o‘tishi mumkin. Odatda, trigger ikkita chiqish yo‘liga ega: birinchisi to‘g‘ri chiqish yo‘li deyiladi va Q bilan belgilanadi. Ikkinchi chiqish yo‘li teskari chiqish yo‘li deyiladi va $\bar{Q}$ bilan belgilanadi.

Triggerning kirish yo‘llarining soni uning bajariladigan funksiyasiga bog‘liqdir. Axborotlarni yozish bo‘yicha triggerlar ikkiga bo‘linadi:

– asinxron triggerlar;

– sinxron triggerlar.

Asinxron triggerlarda axborotning o‘zgarishi uchun istalgan paytda kirish signallarining berilishi kifoya qiladi.

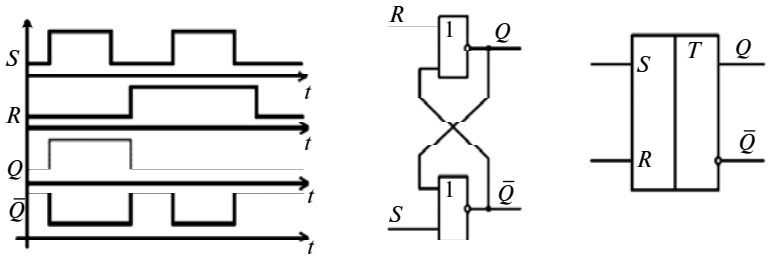
Sinxron triggerlarning chiqish yo‘lida signallarning o‘zgarishi uchun uning kirish yo‘liga qo‘shimcha sinxrosignalni (boshqarish signali) berish kerak. Aks holda bunday triggerlarda axborotlarni yozib bo‘lmaydi.

EHM qurilmalarida turli funksional imkoniyatlarga ega bo‘lgan triggerlar qo‘llaniladi. Lekin barcha sxemalarning asosini asinxron RS-trigger tashkil etadi.

Asinxron RS-trigger. Asinxron RS-trigger ikkita mantiqiy elementlar asosida qurilishi mumkin: YOKI-EMAS va HAMDA-EMAS. Elementlar qayta aloqa zanjirlari orqali o‘zaro ulanadi.

YOKI-EMAS elementlar asosida qurilgan asinxron RS-triggerning funksional sxemasi grafik belgilanishi va uning vaqt diagrammasi 4.1-rasmdagi ko‘rinishga ega.

Bunday trigger ikkita kirish yo‘liga (S va R) hamda ikkita chiqish yo‘liga (Q va $\bar{Q}$) ega. Bu yerda R – reset – olib tashlash



4.1-rasm. Asinxron RS-trigger.

va S – *Set* – o‘rnatish ma’nolarini anglatadi. Asinxron RS-triggerning ishlash prinsipining analitik tahlili quyidagicha:

1) $S = 1$ va $R = 0$ bo‘lganda triggerga 1 yoziladi ($Q = 1$);

2) $S = 0$ va $R = 1$ bo‘lganda triggerga 0 yoziladi ($Q = 0$);

3) $S = 0$ va $R = 0$ bo‘lganda trigger o‘z holatini saqlaydi. Ushbu holat axborotni saqlash rejimi hisoblanadi;

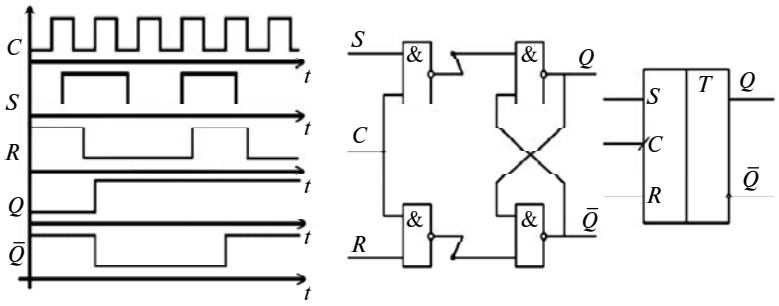
4) $S = 1$ va $R = 1$ signallarini bir vaqtda berish taqiqlanadi, chunki bunday kirish signallarida trigger o‘z turg‘un holatini yo‘qotadi.

Demak, asinxron RS-triggerning o‘tish jadvalini quyidagicha ifodalash mumkin (20-jadval).

20-jadval

S	R	Q_{t+1}	Izoh
0	0	Q_t	Saqlash
0	1	0	0 ni o‘rnatish
1	0	1	1 ni o‘rnatish
1	1	–	Taqiqlangan

Bir taktli sinxron RS-trigger. Mantiqiy elementlarning yoki qurilmalarning kirish yo‘llariga signallar doim ham bir vaqtning o‘zida yetib kelmaydi. Chunki ungacha signallar bir necha elementlardan o‘tib keladi, shuningdek, signallarni bir xil ushlamaydigan elementlar orqali o‘tib keladi. Bunday hodisa signallar poygasi deyiladi. Signallar poygasi natijasida mantiqiy elementlar va qurilmalarning yangi signallari qiymatlari eski signallar qiymatlari bilan qo‘shilib ketishi natijasida mantiqiy element yoki qurilmaning noto‘g‘ri ishlashiga olib kelishi mumkin.



4.2-rasm. HAMDA-EMAS mantiqiy elementlardan tashkil topgan sinxron RS-triggerning vaqt diagrammasi, funksional sxemasi va shartli belgilanishi.

Bunday salbiy hodisalarning oldini olish uchun elementlarning kirish yo‘liga axborot signallardan tashqari qo‘shimcha sinxron (taktli yoki boshqaruvchi) signal beriladi. Chunki sinxron signal berilgunga qadar elementlarning kirish yo‘liga axborot signallari yetib keladi. Sinxron signal esa ularni yozish uchun ruxsat beruvchi boshqaruvchi signal vazifasini bajaradi.

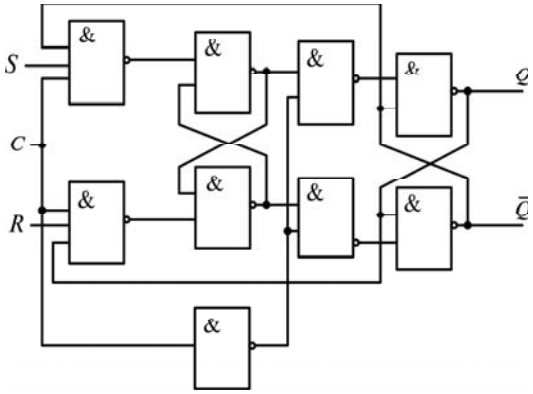
Bir taktli sinxron RS-trigger *R* va *S* axborot kirish yo‘llaridan tashqari qo‘shimcha *C* sinxron kirish yo‘liga ega (*C* – *clock* – asosiy sinxronizatsiya ma’nosini anglatadi). 4.2-rasmda HAMDA-EMAS mantiqiy elementlardan tashkil topgan sinxron RS-triggerning vaqt diagrammasi, funksional hamda prinsipial sxemalari keltirilgan.

Bir taktli sinxron RS-trigger quyidagi prinsip asosida ishlaydi: $C_t = 0$ bo‘lganda trigger o‘z holatini saqlaydi. $C_t = 1$ bo‘lganda, ya’ni triggerlarning kirish yo‘liga sinxrosignal (boshqaruvchi signal) berilganda, u asinxron RS-triggerga o‘xshab ishlaydi.

Demak, sinxron RS-trigger uchun quyidagi o‘tish jadvali o‘rinli.

21-jadval

S	***S***	***R***	Q_{t+1}	**Izoh**
0	*	*	Q_t	Saqlash
1	0	0	Q_t	Saqlash
1	0	1	0	0 ni o‘rnatish
1	1	0	1	1 ni o‘rnatish
1	1	1	–	Taqiqlangan



4.3-rasm. Ikki taktli sinxron RS-triggerning funksional sxemasi.

Ikki taktli sinxron RS-trigger. Bir taktli triggerlarning ishonchli ishlashi uchun unga axborotni yozishdan oldin uning tarkibidagi ikkilik kod boshqa triggerga uzatilgan bo‘lishi kerak. Buning uchun bir-biriga sinxronika bo‘yicha qarama-qarshi bo‘lgan ikkita sinxron RS-triggerdan iborat bo‘lgan ikki taktli sinxron RS-trigger qo‘llaniladi.

Ikki taktli sinxron RS-triggerning funksional sxemasi 4.3-rasmdagi ko‘rinishga ega.

Sxema ikkita bir taktli RS-triggerlardan tashkil topgan. Triggerning kirish yo‘liga $S = 1$ signal berilganda axborot, ya’ni ikkilik kod (0 yoki 1) birinchi triggerga yoziladi. Ikkinchi trigger esa o‘z holatini o‘zgartirmaydi.

$S = 0$ bo‘lganda birinchi trigger axborotni saqlash rejimiga o‘tadi. Ikkinchi trigger esa birinchi triggerning qiymatini qabul qiladi. Chunki, $S = 0$ signal pastki invertordan o‘tib, $S = 1$ qiymatga ega bo‘ladi va ikkinchi triggerga birinchi triggerning qiymatini qabul qilishga ruxsat beradi.

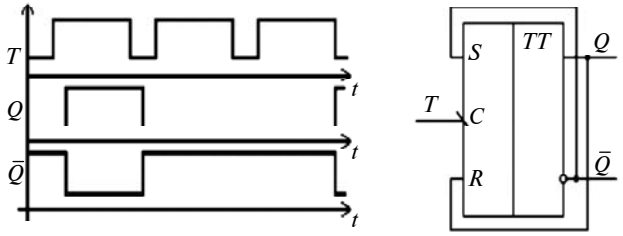
T-trigger. T-trigger (ing. *toggle*) relaksator ma’nosini anglatadi. Ushbu trigger faqat bitta T-axborot kirish yo‘liga ega. Bu kirish yo‘li hisob kirish yo‘li deyiladi. T-trigger o‘z holatini kirish yo‘liga yangi boshqaruvchi signal kelishi bilan o‘zgartiradi.

Asinxron T-trigger. Bu qurilma ikkita turg‘un holatga ega bo‘lib, bitta T-axborot kirish yo‘liga egadir. U asinxron T-trigger deb ataladi. Bunday triggerning kirish yo‘liga 1 signal berilganda u o‘z holatini qarama-qarshi (teskari) holatga o‘zgartiradi. Asinxron T-triggerning ishlash prinsipi 22-jadvalda keltirilgan.

22-jadval

R	Q_{t+1}	Izoh
0	Q_t	Saqlash
1	$\underline{Q_t}$	Inversiya

4.4-rasmda asinxron T-triggerning vaqt diagrammasi hamda ikki taktli RS-triggerdan tashkil topgan sxemasi keltirilgan. Ushbu sxemada S kirish yo‘liga $T = 1$ signalini berish orqali, ikki taktli RS-triggerga oldingi taktga teskari bo‘lgan ikkilik kodni yozish imkoniyati paydo bo‘ladi.



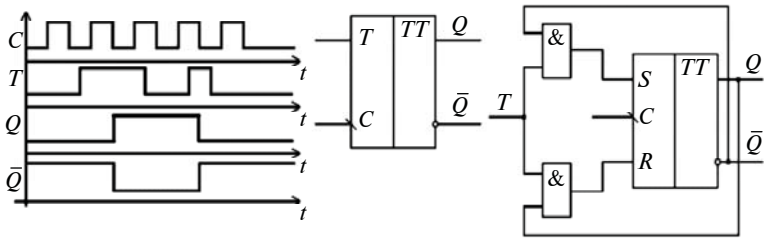
4.4-rasm. Asinxron T-triggerning vaqt diagrammasi va sxemasi.

Sinxron T-trigger. Sinxron ikki taktli ushbu qurilmada birlik kodni triggerga yozish $S = 1$ bo‘lganda bajariladi. T kirish signali $S = 1$ bo‘lganda yuqori kuchlanish bilan ifodalanadi. Demak, triggerning holati $T = 1$ bo‘lganda teskarisiga o‘zgaradi va $T = 0$ bo‘lganda esa uning holati o‘zgarmaydi. Sinxron T-triggerning ishlash prinsipi 23-jadvalda keltirilgan.

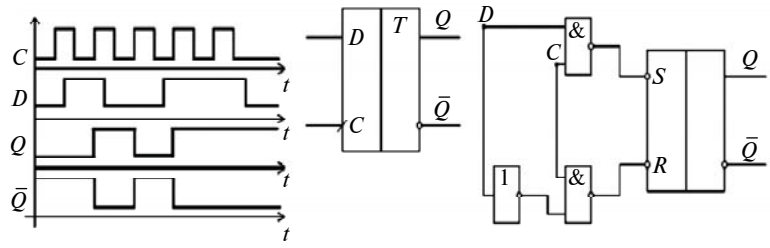
23-jadval

S	R	Q_{t+1}	Izoh
0	*	Q_t	Saqlash
1	0	Q_t	Saqlash
1	1	Q_t	Inversiya

D-trigger. D-triggerlar (ing. *delay* – ushlash) bitta D axborotli kirish yo‘liga ega bo‘lib, vaqtincha signallarni saqlab turish uchun xizmat qiladi. Trigger bitta kirish yo‘liga ega va u ikkita turg‘un holatning birida (0 yoki 1) bo‘lishi mumkin. Bunday trigger bir taktli va ikki taktli bo‘lishi mumkin.



4.5-rasm. Sinxron T-triggerning vaqt diagrammasi, shartli belgilanishi va sxemasi.

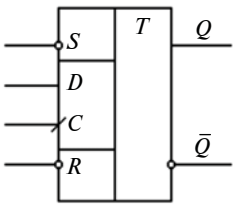


4.6-rasm. D-triggerning vaqt diagrammasi, shartli belgilanishi va sxemasi.

24-jadvalda D-triggerning o‘tish jadvali, 4.6-rasmda uning vaqt diagrammasi, shartli belgilanishi va sxemasi keltirilgan.

24-jadval

S	***R***	Q_{t+1}	**Izoh**
0	*	Q_t	Saqlash
1	0	0	0 ni o‘rnatish
1	1	1	1 ni o‘rnatish



4.7-rasm. Universal D-trigger.

D-triggerni universal trigger sifatida ham qo‘llash mumkin. Unda u ham RS-trigger vazifasini, ham dinamik D-trigger vazifasini bajaradi. Bunday triggerning shartli belgilanishi 4.7-rasmdagi ko‘rinishga ega.

Sxemada *S* va *R* da signal 1 ga teng bo‘lgan vaqtda u *D* va *S* kirish yo‘llariga ega bo‘lgan universal dinamik D-trigger kabi ishlaydi. Trigger quyidagi qonuniyat asosida ishlaydi (25-jadval).

25-jadval

C	***D***	***S***	***R***	Q_{t+1}	**Izoh**
0	*	0	0	–	Taqiqlangan
0	*	0	1	1	1 ni o‘rnatish
0	*	1	0	0	0 ni o‘rnatish
0	*	1	1	Q_t	Saqlash
1	0	*	*	0	0 ni o‘rnatish
1	1	*	*	1	1 ni o‘rnatish

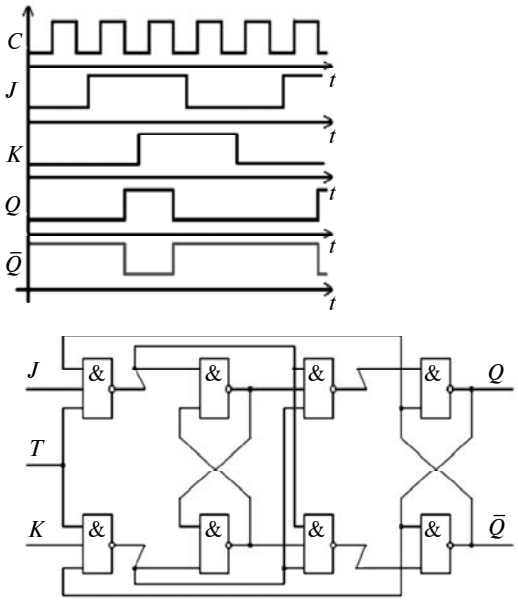
JK-triggerlar. JK-triggerlarning (ing. *jerk* – tezkor yoqish, *kill* – tezkor o‘chirish) RS-triggerdan farqi shundaki, agar RS-triggerlarning kirish yo‘liga $R = 1$ va $S = 1$ signallarini berish taqiqlangan bo‘lsa, JK-triggerlarning kirish yo‘liga bunday signallarni berish mumkin va unda ular o‘z holatini teskarisiga o‘zgartiradi. 26-jadvalda JK-triggerning o‘tish qonuniyati, 4.8-rasmda vaqt diagrammasi va funksional sxemasi keltirilgan.

26-jadval

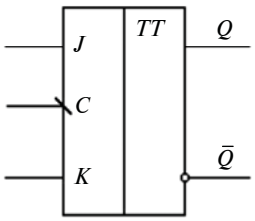
S	***J***	***K***	Q_{t+1}	**Izoh**
0	*	*	Q_t	Saqlash
1	0	0	Q_t	1 ni o‘rnatish
1	1	0	1	1 ni o‘rnatish
1	0	1	0	0 ni o‘rnatish
1	1	1	Q_t	Inversiya

Sxemada «S» kirish yo‘liga impulsning oldingi fronti berilsa, unda birinchi sinxron RS-trigger ishlaydi. RS-triggerning chiqish yo‘lidagi qiymatlari *J* va *K* kirish yo‘llari qiymatlari bilan belgilanadi. Ikkinchi RS-trigger saqlash rejimida bo‘ladi. *S* kirish yo‘liga sinxroimpulsning orqa fronti berilsa, birinchi RS-trigger ikkilik kodni saqlash rejimiga o‘tadi. Uning chiqish yo‘lidagi qiymati ikkinchi RS-triggerga o‘tadi.

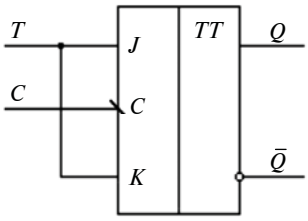
Raqamli texnikada universal JK-triggerlar ham keng qo‘llaniladi. Ular D, T va RS-triggerlar sifatida ishlatilishi va ularning funksiyalarini bajarishi mumkin. 4.9–4.12-rasmlardagi sxemalarda JK-triggerni D, T va RS-triggerlar sifatida qo‘llashning sxematik imkoniyatlari keltirilgan.



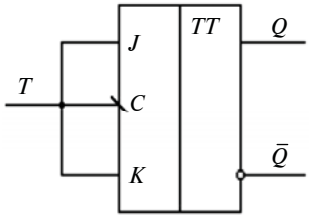
4.8-rasm. JK-triggerning vaqt diagrammasi va funksional sxemasi.



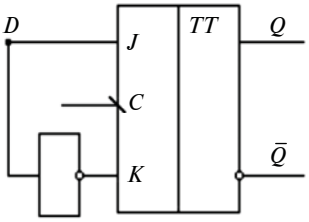
4.9-rasm. Sinxron RS-trigger.



4.10-rasm. Sinxron T-trigger.



4.11-rasm. Asinxron T-trigger.



4.12-rasm. Sinxron D-trigger.

Nazorat savollari va topshiriqlari

1. Sinxron RS-triggerning vaqt diagrammasini quring.
2. D va JK-triggerlar nima bilan farq qiladi?
3. Ikki taktli RS-triggerning qanday afzallik tomonlari bor?
4. JK-triggerning vaqt diagrammasini tushuntiring.
5. D-triggerning funksional vazifalarini belgilang.

4.2. REGISTRLAR

Registrlar – axborotlarni saqlash va ular ustida ayrim amallarni bajarish uchun xizmat qiladigan EHMlarning uzeli yoki operatsion elementidir.

Registrlar, odatda, triggerlar asosida quriladi. Triggerlarning soni registrning razryadiga bog‘liq.

Registrlar quyidagi amallarni bajaradi:

1) registrni nol yoki bir (0 yoki 1) holatga o‘rnatish;
2) boshqa qurilmadan kodni qabul qilish;
3) boshqa qurilmaga kodni uzatish;
4) kerakli razryad sonigacha kodni chapga yoki o‘ngga siljitish;
5) ketma-ket kodni parallel kodga o‘zgartirish va aksincha;
6) to‘g‘ri kodni teskari yoki qo‘shimcha kodga o‘zgartirish va aksincha;
7) mantiqiy amallarni bajarish.

Registrlar parallel, ketma-ket, ketma-ket-parallel va parallel-ketma-ket turlarga bo‘linadi.

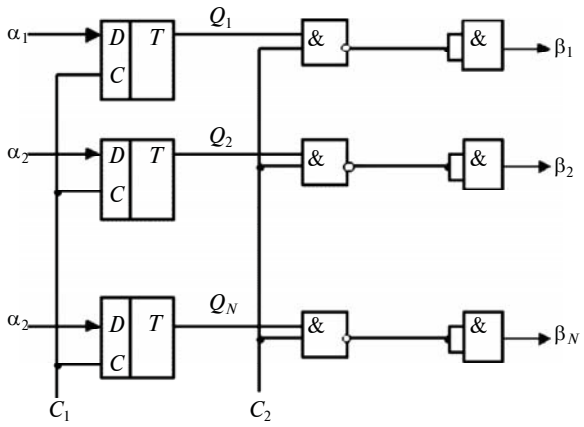
Yangi axborotni registrga kiritish yozish amali deyiladi. Axborotni registrdan chiqarib boshqa uzelga uzatish o‘qish rejimi deyiladi. Ayrim hollarda registrning barcha triggerlarini bitta boshqaruvchi signaldan foydalanib ma’lum holatga (masalan, nol holatga) o‘tkazish talab etiladi. Ushbu operatsiya boshlang‘ich holatga o‘tkazish deb nomlanadi.

Parallel registrlar. Parallel registrlarni D yoki RS-triggerlar asosida qurish mumkin. 4.13-rasmda D-trigger asosida qurilgan parallel registrning sxemasi keltirilgan. Unda S_1 kirish yo‘liga axborotni yozish signallari va S_2 kirish yo‘liga esa o‘qish signallari beriladi. α – registrga yoziladigan (axborotlar) ikkilik kodlar. Sxema quyidagi qonuniyatlar asosida ishlaydi:

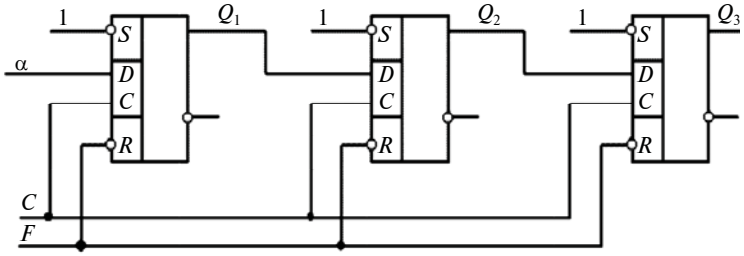
$C_1^n = C_2^n = 0$ bo‘lganda $Q_i^n = Q_i^{n-1}$, $\beta_i^n = 0$;

$C_1^n = 0$; $C_2^n = 1$ bo‘lganda $Q_i^n = Q_i^{n-1}$, $\beta_i^n = Q_i^n$;

$C_1^n = 0$; $C_2^n = 0$ bo‘lganda $Q_i^n = x_i^{n-1}$, $\beta_i^n = 0$.



4.13-rasm. D-trigger asosida qurilgan parallel registr.



4.14-rasm. DRS-triggerlar asosida qurilgan ketma-ket registr.

$C_1^n \cdot C_2^n \neq 1$, ya'ni bir vaqtning o'zida axborotni ham o'qish, ham yozish mumkin emas.

Ketma-ket registrlar. DRS-triggerlar asosida qurilgan ketma-ket registrning sxemasi 4.14-rasmda keltirilgan.

Registrning S kirish yo'liga axborotlarni yozish va o'qish signallari beriladi, F kirish yo'liga esa 0 ni o'rnatish signali beriladi.

Ushbu sxemaning ishlash tartibi quyidagicha:

α va S kirish signallarining qiymatidan qat'i nazar $F^n = 0$ bo'lganda $Q_1^n = Q_2^n = Q_3^n = \beta^n = 0$ bo'ladi.

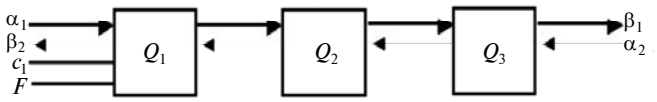
$F^n = 1$ bo'lganda quyidagi vaziyatlar bo'lishi mumkin:

1) $C^n = 0;\ Q_1^n = Q_1^{n-1}; Q_2^n = Q_2^{n-1}; Q_3^n = Q_3^{n-1} = \beta^n;$

2) $C^n = 0;\ Q_1^n = Q_1^{n-1}; Q_2^n = Q_2^{n-1}; Q_3^n = Q_3^{n-1} = \beta^n.$

I z o h : DRS-trigger – D-trigger bo‘lib, unda nolni (*R* kirish yo‘li) va birni (*S* kirish yo‘li) o‘rnatish uchun qo‘shimcha kirish yo‘llari qo‘llanilgan.

Reversiv registr. Reversiv registrlar axborotni ham o‘ng yo‘nalish bo‘yicha, ham chap yo‘nalish bo‘yicha siljitib borish imkoniyatiga ega. 4.15-rasmda uch razryadli reversiv registrning sxemasi keltirilgan. U uchta blokdan tashkil topgan.



4.15-rasm. Uch razryadli reversiv registrning sxemasi.

$S_1^n = 1,\ F^n = 0$ bo‘lganda axborot chapdan o‘ng tomonga qarab siljitiladi, $S_1^n = 1,\ F^n = 1$ bo‘lganda axborot o‘ngdan chapga qarab siljiydi.

Q_2 «o‘rta blokning» ishlash tartibi quyidagi ko‘rinishga ega:

agar $S_1^n = F^n = 0$ bo‘lsa, u holda $Q_2^n = Q_2^{n-1}$;

agar $S_1^n = 1,\ F^n = 0$ bo‘lsa, u holda $Q_2^n = Q_1^{n-1}$;

agar $S_1^n = 0,\ F^n = 1$ bo‘lsa, u holda $Q_2^n = Q_2^{n-1}$;

agar $S_1^n = F^n = 1$ bo‘lsa, u holda $Q_2^n = Q_3^{n-1}$.

Sxemada «chekka» bloklarning ishlash tartibi «o‘rta» bloknikiga deyarli o‘xshash. Faqat ularning farqi shundaki, Q_1 blokka α_1 kirish signali kiradi va undan β_2 chiqish signali olinadi, Q_3 blokka esa α_2 kirish signali kiradi va undan β_1 chiqish signali olinadi.

Nazorat savollari va topshiriqlari

1. Registr iborasiga ta’rif bering.
2. Uch razryadli reversiv registrning ishlash prinsipini tushuntiring.
3. Registr qanday vazifalarni bajaradi?
4. Ketma-ket registrning ishlash prinsipini tushuntiring.
5. Parallel registrning ishlash prinsipini tushuntiring.

4.3. HISOBLAGICHLAR

Hisoblagichlar ketma-ket qurilma bo‘lib, kirish yo‘liga berilgan impulslar sonini hisoblash va ularni saqlash uchun xizmat qiladi.

Hisoblagichlar quyidagi belgilar bo‘yicha tasniflanadi:

Hisob yo‘nalishi bo‘yicha: yig‘iluvchi; ayiruvchi; reversiv.

O‘tish sxemalarini tashkil etish yo‘li bo‘yicha: ketma-ket o‘tuvchi hisoblagichlar; to‘g‘ridan to‘g‘ri o‘tuvchi hisoblagichlar; parallel o‘tuvchi hisoblagichlar; kombinatsiyalashgan o‘tuvchi hisoblagichlar.

O‘tish koeffitsiyenti bo‘yicha: ikkilik; ikkilik-o‘nlik; ixtiyoriy doimiy qayta hisoblash koeffitsiyenti; hisobning o‘zgaruvchan koeffitsiyenti hisoblagichlari va h.k.

Hisoblagichning sinxronizatsiya asosida holatini o‘zgartirishi bo‘yicha: asinxron; sinxron hisoblagichlar.

Har qanday chekli avtomatlar kabi hisoblagichning istalgan vaqtdagi holati uning triggerlarida tayinlangan Q kod bilan aniqlanadi. Hisoblagichning boshlang‘ich holati Q^0 bilan belgilanadi. Uning kirish yo‘liga n ta o‘tish signali berilganda hisoblagichning holati Q^n bilan belgilanadi. Umuman olganda, hisoblagichning ishlash tartibini belgilash $Q = f(n)$ funksiyani biror usul yordamida aniqlash demakdir. Ushbu funksiyada: $n = 0, 1, 2,..., k$, bu yerda n – kirishdagi o‘tish signali tartibi. Tabiiyki, istalgan n da $Q^n = Q^{n+k}$ bo‘ladi.

Har qanday o‘tishni tashkil etishdan qat’i nazar N-razryadli ikkilik jamlovchi hisoblagichning ishlash tartibi quyidagi ko‘rinishga ega:

$$Q^n = \begin{cases} Q^{n-1} + 1, \text{ bunda } Q^{n-1} \neq 2^N - 1; \\ 0 \quad\quad , \text{ bunda } Q^{n-1} = 2^N - 1. \end{cases}$$

27-jadval

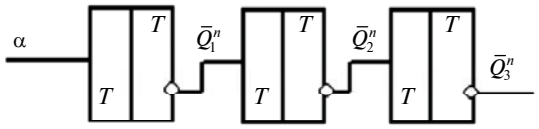
Q^n			
n	Q_3^n	Q_2^n	Q_1^n
0	0	0	0
1	0	0	1
2	0	1	0
3	0	1	1
4	1	0	0
5	1	0	1
6	1	1	0
7	1	1	1

N kichik bo‘lganda hisoblagichning ishlash tartibi o‘tish jadvali ko‘rinishida beriladi. Masalan, $N = 3$ va $Q^0 = 000$ bo‘lganda bu 27-jadvaldagi ko‘rinishga ega.

Hisoblagichning barcha triggerlari bir xil deb hisoblagan taqdirda, uni o‘qish jadvali asosida sintez qilish qulaydir. Darhaqiqat, hisoblagichning kichik razryadiga tegishli tayinlangan sonning Q_1^n ustunidan ko‘rinib turibdiki, hisoblagichning birinchi triggeri T-triggerdir. Chunki har bir kirish signali bu triggerni yangi holatga o‘tkazadi. Demak, boshqa triggerlar ham T-triggerlar bo‘lib hisoblanadi.

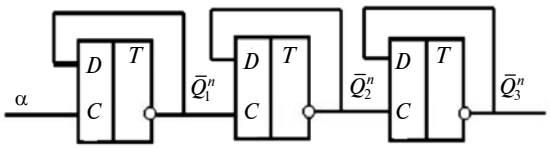
Ketma-ket o‘tishga asoslangan hisoblagichni sintez qilamiz. Bunday hisoblagichlarda har bir trigger keyingi trigger uchun o‘tish signalini ishlab chiqadi. O‘tish jadvalidan ko‘rinib turibdiki, i-triggerning yangi holatga o‘tishi, $(i-1)$-triggerning 1 holatdan 0 holatga o‘tishi bilan amalga oshiriladi. Shuning uchun $(i-1)$-triggerning inversli chiqish qiymatini i-triggerning T kirish yo‘li bilan bog‘lash kerak.

Bunday hisoblagichning sxemasi 4.16-rasmdagi ko‘rinishga ega.



4.16-rasm. T-triggerlar asosida qurilgan ketma-ket o‘tishga asoslangan hisoblagich.

Shu tariqa ketma-ket o‘tuvchi jamlovchi hisoblagichning sxemasini D-triggerlar asosida qurish mumkin. Bunday hisoblagichda har bir D-trigger T-trigger kabi ishlaydi. Bunday sxema 4.17-rasmdagi ko‘rinishga ega.



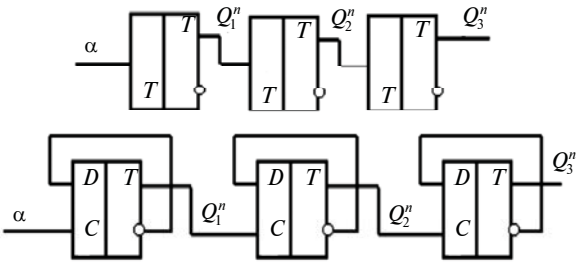
4.17-rasm. D-triggerlar asosida qurilgan ketma-ket o‘tuvchi jamlovchi hisoblagich.

O‘tishning qanday tashkil etilishidan qat’i nazar N-razryadli ikkilik ayiruvchi hisoblagichning ishlash tartibi quyidagicha:

$$Q^n = \begin{cases} Q^{n-1} - 1, & \text{bunda } Q^{n-1} \neq 0; \\ 2^N - 1, & \text{bunda } Q^{n-1} \neq 0. \end{cases}$$

Oldingi o‘tkazilgan tahlillar kabi fikr yuritilganda ketma-ket o‘tuvchi 4.18-rasmdagi ayiruvchi hisoblagichlarning sxemalarini hosil qilish mumkin.

Agar ko‘rilgan har qanday hisoblagichlarda i-triggerning chiqish yo‘li bilan $(i+1)$-triggerning kirish yo‘li orasida kommutatsiya



4.18-rasm. T va D-triggerlar asosida qurilgan ayiruvchi hisoblagichlar.

sxemasi kiritilganda va unga qo‘shimcha F boshqaruvchi signal berilganda, ketma-ket o‘tuvchi reversiv hisoblagichning sxemasini hosil qilamiz.

Aytaylik, $F^n = 0$ bo‘lganda reversiv hisoblagich yig‘uvchi sifatida va $F^n = 1$ bo‘lganda ayiruvchi sifatida ishlaydigan bo‘lsin. U holda kommutatsiya sxemasining ishlash tartibi quyidagicha bo‘lishi kerak:

$$T_i = \begin{cases} \overline{Q}_{i-1}, & \text{bunda} \quad F = 0; \\ Q_{i-1}, & \text{bunda} \quad F = 1, \end{cases}$$

bu yerda: T_i – signal (T kirish yo‘lida).

Lekin F boshqaruv signali qiymatining o‘zgarishi kirish signali berilishi bilan hisoblagich holatining o‘zgarishiga olib kelishi mumkin. Noto‘g‘ri o‘tishlarning oldini olish maqsadida F signal faqat $\alpha = 0$ bo‘lganda uning o‘zgarishini va shu asosda kommutatsiya sxemasining ishlash tartibida quyidagi o‘zgartirishni qabul qilish mumkin:

$$T_i = \begin{cases} 0, & \alpha = 0, \\ \overline{Q}_{i-1}, & \alpha = 1, F = 0, \\ Q_{i-1}, & \alpha = 1, F = 1. \end{cases}$$

Bu tartib quyidagi uyg‘onish funksiyasini aniqlaydi:

$$T_i = f(\alpha, F, Q_{i-1}).$$

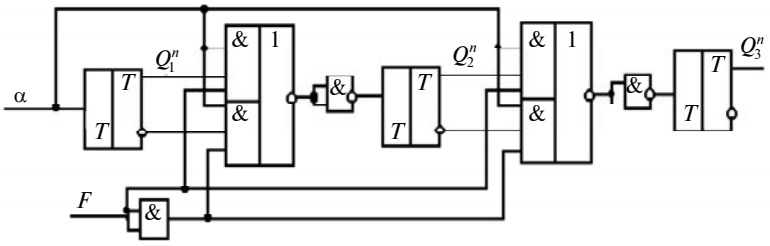
Unga 28-jadval mos keladi. Bu jadvaldan quyidagini hosil qilamiz:

$$T_i = \alpha \overline{F} \overline{Q}_{i-1} + \alpha F Q_{i-1} = \overline{\overline{\alpha \overline{F} \overline{Q}_{i-1} + \alpha F Q_{i-1}}}.$$

Uch razryadli ketma-ket o‘tuvchi reversiv hisoblagichning sxemasi 4.19-rasmdagi ko‘rinishga ega bo‘ladi.

28-jadval

A	***F***	Q_{i-1}	T_i
0 } taqiq	0	0	0
0 } taqiq	0	1	0
0 } taqiq	1	0	0
0 } taqiq	1	1	0
1 } hisob	0 } qo‘shish	0 } $\bar{\alpha}$ →	1
1 } hisob	0 } qo‘shish	1 } $\bar{\alpha}$ →	0
1 } hisob	1 } ayirish	0 } α →	0
1 } hisob	1 } ayirish	1 } α →	1



4.19-rasm. Uch razryadli reversiv hisoblagich.

Nazorat savollari va topshiriqlari

1. Hisoblagichning qanday turlarini bilasiz?
2. Sinxron va asinxron hisoblagichlar bir-biridan nima bilan farq qiladi?
3. Hisoblagich qanday elementlar asosida quriladi?
4. Reversiv hisoblagichning ishlash prinsipini tushuntiring.

maksimal xotira soni manzil (adres) maydoni deyiladi. Manzil maydoni manzilli shinaning razryadliligi bilan aniqlanadi.

Mikroprotsessorli tizimlarning tahlilini oddiy (bazaviy) modellardan boshlash maqsadga muvofiqdir. Ular mikroprotsessorning tashkiliy strukturasini, uning buyruqlari tizimini va quyi tizim faoliyatini sodda va tushunarli tarzda akslantiradi. Bundan tashqari, mikroprotsessor tizimlarining kichik avlodida joriy etilgan ko‘plab prinsipial yechimlar uning keyingi katta modellarida ham saqlangan. Yana shuni ta’kidlash joizki, mikroprotsessorlarning eng oxirgi modellarining tashkiliy strukturasi hamda texnologiyasi ularni ishlab chiqqan firmalar tomonidan «nou-xau» sifatida targ‘ibot qilinmaydi.

Mikroprotsessor tizimlarining tashkiliy strukturasini nisbatan sodda bo‘lgan K1810VM86 mikroprotsessori (i8086) misolida quyidagi quyi tizimlarni ajratgan holda ko‘rib chiqamiz:

- protsessorli modul;
- xotira;
- kiritish/chiqarish;
- uzilishlar;
- xotiraga to‘g‘ridan to‘g‘ri murojaat.

Nazorat savollari va topshiriqlari

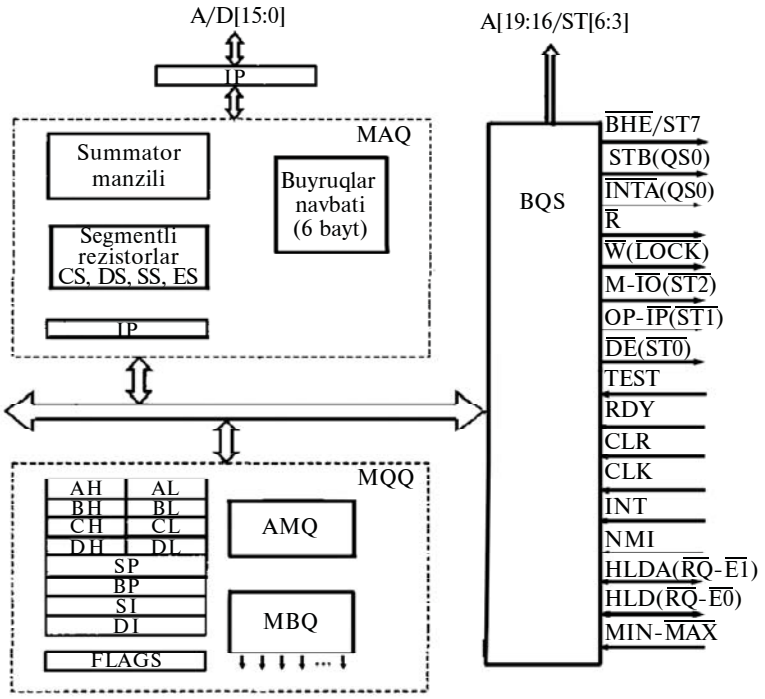
1. Mikroprotsessor qanday vazifalarni bajarishga xizmat qiladi?
2. Mikroprotsessorning asosiy xarakteristikalariga nimalar kiradi?
3. Razryadlilik deganda nimani tushunasiz?
4. Arifmetik-mantiqiy qurilmaning vazifalarini belgilang.

5.2. MIKROPROTSESSORNING ICHKI STRUKTURASI

Protsessorli modul har qanday mikroprotsessorli tizimning asosiy qismini tashkil etadi. U mikroprotsessorning o‘zidan tashqari bir qator yordamchi sxemalardan tashkil topgan bo‘lib, ular bilan hamkorlikda mikroprotsessor o‘z funksiyasini bajaradi. Bunday yordamchi sxemalarga taktli generator, interfeysli sxemalarni misol sifatida keltirish mumkin.

K1810VM86 (i8086) mikroprotsessorining strukturali sxemasi 5.1-rasmda keltirilgan. Ushbu mikroprotsessor quyidagi uchta asosiy qurilmadan tashkil topgan:

- MQQ – ma’lumotlarni qayta ishlash qurilmasi;
- MAQ – magistral bilan aloqa qurilmasi;
- BSQ – boshqarish va sinxronizatsiya qurilmasi.



5.1-rasm. Mikroprotsessorning strukturasi.

MQQ buyruqlarni bajarish uchun xizmat qiladi. U 16-razryadli arifmetik-mantiqiy qurilma (AMQ), tizimli registrlar va boshqa yordamchi sxemalar, shuningdek, registrlar bloki (bazali va indeksli umumiy vazifali registrlar) hamda mikrodasturli boshqarish blokidan tashkil topgan.

MAQ xotiraning 20 razryadli fizik manzilini va 16 razryadli tashqi qurilma manzilini shakllantirishni, xotiradan buyruqlarni tanlashni, shuningdek, xotira qurilmalari, tashqi qurilmalar va magistral bo‘yicha boshqa protsessorlar bilan aloqadorlikni ta’minlaydi. MAQ o‘zida summator manzilini, buyruqlar navbati registri blokini va segmentli registrlar blokini mujassamlashtirgan.

BSQ mikroprotsessor qurilmalari ishining sinxronizatsiyasini, boshqaruv signallarini ishlab chiqishni va boshqa qurilmalar bilan aloqa o‘rnatish uchun holat signallarini, shuningdek, mikroprotsessorli tizimning boshqa qurilmalari holatining tahlili va ularning signalga bo‘lgan ta’sirini ta’minlaydi.

K1810VM86 (i8086) mikroprotsessori minimal va maksimal rejimlarda ishlashi mumkin.

Minimal rejim mikroprotsessorli tizimning bir kristalli konfiguratsiyasini tashkil etish uchun mo‘ljallangan. Maksimal rejim esa tizimda umumiy tizimli shina uchun ishlaydigan bir nechta mikroprotsessorlarning bo‘lishini nazarda tutadi.

Mikroprotsessorning tashqi chiqish yo‘llarida signallarni multipleksorlashtirish prinsipi keng qo‘llaniladi. Unga ko‘ra vaqt bo‘yicha taqsimlangan turli signallar umumiy yo‘llar orqali uzatiladi. Bundan tashqari, ish rejimidan (min — max) kelib chiqqan holda, ma’lum chiqish yo‘llaridan turli signallarni uzatish uchun foydalanish mumkin.

29-jadvalda K1810VM86 (i8086) mikroprotsessorlari tashqi chiqish yo‘llarining tavsifi keltirilgan. Chiqish yo‘llarini tavsiflashda og‘ma chiziq bilan (/) belgilangan signallar mashina siklining turli vaqtida paydo bo‘ladi. Kichik qavs ichidagi signallar faqat maksimal rejimga xosdir. Jadvalda signaldan keyingi * simvoli uning irversiya belgisini bildiradi.

29-jadval

Tashqi chiqish yo‘li	**Tavsifi**
A/D[15:0]	Manzil/ma’lumotlarning kichik 0–15 razryadlari.
A[19:16]/ST[6:3]	Manzil/signallar holatining 16–19 razryadlari.
VNE* /ST[7]	Ma’lumotlar/ holat signali katta baytini uzatishga ruxsat.
STB(QSO)	Manzil strobi (buyruqlar navbatining holati).
R*	O‘qish.
W*/(LOCK*)	Yozish (kanalni blokirovkalash).
M-IO*(ST2*)	Xotira – tashqi qurilma (sikl holati).
OP-IP*(STl*)	Uzatish/qabul qilish (sikl holati).
DE*(STO*)	Ma’lumotlarni uzatishga ruxsat (sikl holati).
TEST*	Tekshirish.
RDY	Tayyorlik.
CLR	Bekor qilish.
CLC	Taktli signal.
INT	Tashqi uzilishning so‘rovi.
INTA*(QS1)	Uzilishni tasdiqlash (buyruqlar navbatining holati).
NMI	Niqoblanmaydigan uzilishning so‘rovi.
HLD(RQ*/EO)	PDP so‘rovi. Magistralga ruxsatning so‘rovi/tasdig‘i.
NLDA(RQ*/El)	PDP tasdig‘i. Magistralga ruxsatning so‘rovi/tasdig‘i.
MIN/MAX*	Rejimni berish potensiali (min = 1, max = 0).

Nazorat savollari va topshiriqlari

1. Mikroprotsessorning strukturasi qaysi qurilmalardan tashkil topgan?
2. Mikroprotsessorning ishlash prinsipini tushuntiring.
3. Mikroprotsessorning boshqarish va sinxronizatsiya qurilmasi qanday vazifalarni bajaradi?

5.3. MIKROPROTSESSORNING BUYRUQLI VA MASHINALI SIKLLARI

K1810VM86 (i8086) mikroprotsessorlari mikroprotsessorli tizim tarkibida ikki bayt uzunlikdagi so‘zlar orqali xotira va tashqi qurilmalar bilan axborot almashadi. Chunki uning ma’lumotlar shinasi 6 bitdan iborat. Mikroprotsessor ishining asosini buyruq sikli tashkil etadi. Buyruq sikli xotiradan ma’lum amalni yoki biror boshqa funksiyani tanlash va bitta buyruqni bajarishni anglatadi.

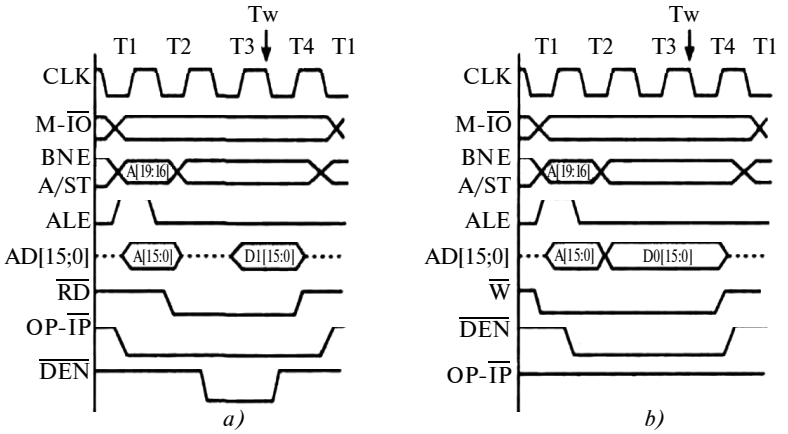
Har qanday buyruq sikli xotiradan buyruqning birinchi so‘zini olishdan boshlanadi. Bu buyruqning manzili buyruq hisoblagichida saqlanadi. K1810VM86 (i8086) mikroprotsessorlari buyruqlarining uzunligi 1 baytdan 6 baytgacha bo‘lishi mumkin. Bunda birinchi so‘zni buyruqning uzunligi haqidagi axborot tashkil etadi. Shunday qilib, xotiradan bitta buyruqni olish uchun tezkor xotiraga bitta yoki bir nechta murojaatlarni tashkil etish talab etiladi.

Buyruq sikli buyruqning turi va formatiga qarab, shuningdek, manzillash usuli va operandalar soniga qarab xotiraga va tashqi qurilmaga turlicha sondagi murojaatlarni mujassamlashtiradi. Chunki buyruq sikli uchun buyruqning o‘zini o‘qishdan tashqari, operandalarni o‘qish va natijalarni joylashtirish ehtiyoji paydo bo‘lishi mumkin.

Xotira qurilmasiga yoki tashqi qurilmaga murojaatlar buyruq siklining turli qismlarida joylashganiga qaramasdan, ular boshqaruvchi avtomatning umumiy jihozlarida mikroprotsessorning interfeysiga mos holda yagona tartib asosida bajariladi. Mikroprotsessorga buyruqning bitta so‘zini, ma’lumotlarni uzatish yoki qabul qilish harakatlari mikroprotsessorli tizimning **mashinali sikli** deyiladi. Buyruq sikli bitta yoki bir nechta mashinali sikllardan tashkil topgan.

Mashinali sikl quyidagi amallarni o‘z ichiga oladi:

- protsessor tomonidan murojaat etiladigan xotira yoki tashqi qurilma manzilini uzatish;
- mashinali sikl turini xarakterlaydigan va ma’lumotlarni uzatish yo‘nalishini (M-IO, OP-IP) boshqarish signallarini berish;
- sinxronlanadigan signallarni (STB, R, W) berish;
- ma’lumotlarni uzatish.



5.2-rasm. Mikroprotsessorning mashinali sikllari:
a) O‘QISH sikli; *b)* YOZISH sikli.

K1810VM86 (i8086) mikroprotsessorlarida manzillar/ma’lumotlarning multipleksorlashgan shinasi amalga oshirilgan. Bu kristallning chiqish yo‘llari sonining noyobligi va manzilni uzatish uchun qo‘shimcha takt talab etilishi hamda A/D umumiy shinada manzilning mavjudligini hosil qilish uchun qo‘shimcha boshqarish signalini (STB) uzatish zaruriyati bilan bog‘liq.

Mashinali siklning turli funksiyalarini, asosan, ikkita ko‘rinishga olib kelish mumkin – o‘qish (ma’lumotlar yoki buyruqlar protsessorda qabul qilinadi) va yozish (ma’lumotlar protsessordan uzatiladi). Mashinali siklning mos ravishdagi bunday jarayonlari 5.2-rasmdagi vaqt diagrammalarda ifodalangan.

Sikl T1 taktda M-IO signalni shakllantirishdan boshlanadi. Bu signal ma’lumotlarning o‘zaro almashuvini amalga oshiradigan qurilmaning turini (xotira, tashqi qurilma) aniqlaydi. M-IO signalining davomiyligi mashinali siklning davomiyligi bilan teng bo‘lib, u qurilma manzili seleksiyasini ta’minlashda foydalaniladi.

T1 da va T2 ning boshlanishida mikroprotsessor A[19:16] va A[15:0] manzillarni va VNE signalini ajratadi. VNE signali AO bilan birgalikda yoki butun so‘zni, yoki uning baytlarining birini uzatishni tanlaydi. ALE signalining pastki frontida tashqi registrlarda manzil shakllanadi. T2 taktda shinalarning o‘tishi kuzatiladi: A[19:16]/ST[6:3] chiqish yo‘llarida holat signali kelib tushadi, A/D[15:0] chiqish yo‘llari esa ma’lumotlarni uzatish/qabul qilish uchun ishlatiladi.

Yuqorida tavsiflangan mashinali sikllar sinxronli hisoblanadi. Lekin bunday axborot almashuvi mikroprotsessorning tezligidan qolishmaydigan qurilmalar bilan amalga oshiriladi. Aks holda mikroprotsessor axborot almashuvining asinxron usuliga o‘tishi kerak. Chunki asinxron usulda mikroprotsessor qurilmadan axborot almashishga tayyor ekanligi yoki axborot almashish jarayoni tugaganligi xususidagi signallarni tahlil qilishi lozim.

Bunday signalning vazifasini K1810VM86 mikroprotsessorida, shuningdek, i8086 va undan katta bo‘lgan keyingi avlod mikroprotsessorlarida RDY (ing. *ready* — tayyor) kirish yo‘li bajaradi. RDY tezkor qurilmalar bilan sinxron axborot almashish uchun doim tayyor bo‘lishi kerak.

Tezkor bo‘lmagan qurilmalar bilan axborot almashishda RDYning faoliyati faol bo‘lmaydi va bu jarayon qurilma va mikroprotsessor orasidagi axborot almashish jarayoni tugaguncha davom etadi.

Qurilmaning tayyor bo‘lishigacha ketadigan vaqtni protsessor uzoq muddat kutishi mumkin.

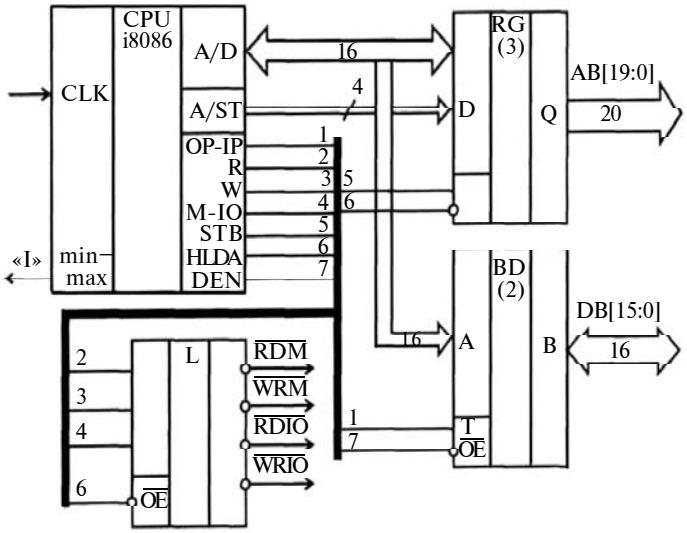
Buning uchun T3 taktda protsessor RDY signalining qiymatini tekshiradi. Agar u faol bo‘lmasa, TZ taktdan keyin mashinali siklda ixtiyoriy sondagi Tw kutish takti qo‘yiladi va bu taktlarning har birida RDY signalining qiymati tekshirib boriladi. RDYning faolligi haqidagi signal paydo bo‘lganda, mikroprotsessor T4 taktga o‘tadi va mashinali siklni tugatadi. Shunday qilib, mikroprotsessor ishini har xil tezlikka ega bo‘lgan qurilmalar bilan kelishtirish imkoniyati yaratiladi.

Nazorat savollari va topshiriqlari

1. Mashinali sikl deb nimaga aytiladi?
2. Mashinali siklda qanday amallar bajariladi?
3. Mikroprotsessorda RDY kirish yo‘lining vazifalarini belgilang.

5.4. MIKROPROTSESSORLI MODULLAR

Ko‘pchilik mikroprotsessorlar mikroprotsessorli tizimlar tarkibida ayrim qo‘shimcha sxemalarsiz ishlay olmaydi. Bunday qo‘shimcha sxemalar mikroprotsessor bilan birgalikda mikroprotsessorli modulni tashkil etadi. Dastavval mikroprotsessorning CLK kirish yo‘liga maxsus tashqi taktli generatordan to‘g‘ri burchakli takt chastotali impulslarni berish zarur.



5.3-rasm. Mikroprotsessorli modulning strukturasi.

K1810VM86 (i8086) mikroprotsessorlari uchun takt impulslari chastotasi 2–6 MHz diapazon oralig‘ida bo‘lishi mumkin.

5.3-rasmda K1810VM86 (i8086) mikroprotsessorlari bazasida yaratilgan protsessorli modulning sodda funksional sxemasining variantlaridan biri keltirilgan. Sxemada ayrim elementlar va aloqalar (masalan, boshlang‘ich holatni olib tashlash sxemasi va h.k.) ko‘rsatilmagan.

Ushbu mikroprotsessor n-MDP-texnologiya asosida ishlab chiqilgan va uning chiqish kaskadlari tizimli interfeys yo‘nalishi uchun yetarli yuklanish qobiliyatini ta’minlay olmaydi. Shuning uchun, mikroprotsessorning chiqish yo‘llariga, odatda, TTL texnologiyalarga asoslangan BD-buferli sxemalar ulanadi.

Bundan tashqari, ushbu mikroprotsessorning manzil va ma’lumotlar shinasi multipleksorlangandir. Mikroprotsessorning chiqish yo‘llarida manzil faqat bitta mashinali sikl takti mobaynida ushlanadi, lekin butun mashinali sikl foydalanilishi kerak bo‘ladi. Shuning uchun manzilni maxsus tashqi RG-registrlarda (u manzil shinasi buferli sxemasi rolini ham bajaradi) eslab qolish zarur.

Shuningdek, ko‘p hollarda mikroprotsessor tomonidan beriladigan boshqarish signallarini tizimli interfeysning standart signallariga o‘zgartirishga to‘g‘ri keladi. Mikroprotsessor mashinali sikl

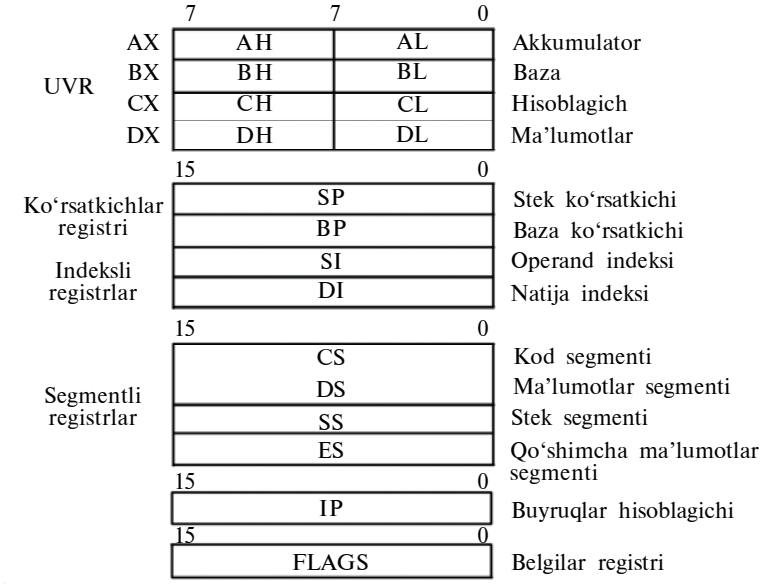
turini identifikatsiyalovchi chiqish signallarini, shuningdek, M-IO, OP-IP, R, W sinxronizatsiyalash (stroblash) signallarini shakllantiradi. Tizimli shina RDM, WRM-xotiraga yozish va xotiradan o‘qish signallaridan, shuningdek, RDIO, WRIO – tashqi qurilmaga yozish va tashqi qurilmaga o‘qish signallaridan foydalanadi. Oddiy hisoblangan L mantiqiy sxema protsessorli signallarni shinali signallarga o‘zgartiradi.

Nazorat savollari va topshiriqlari

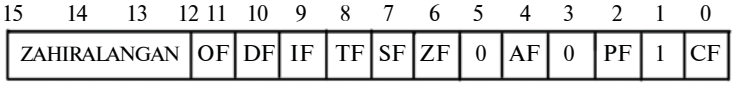
1. Mikroprotsessorli modul nimalardan tashkil topgan?
2. Mikroprotsessorli modulning ishlash prinsipini tushuntiring.

5.5. FOYDALANUVCHI MASHINASI VA BUYRUQLAR TIZIMI

Mikroprotsessorning dasturiy modeli (5.4-rasm) dasturiy-ruxsat berilgan obyektlarni o‘z ichiga oladi va ularning holatini mikroprotsessor buyruqlari yordamida tahlil qilish yoki o‘zgartirish mumkin. Bunday obyektlarga mikroprotsessorning ichki registrlari, xotira yacheykalari va kiritish-chiqarish portlari kiradi.



5.4-rasm. Mikroprotsessor – foydalanuvchi mashinasi.



5.5-rasm. Belgilar registrining formati.

K1810VM86 (i8086) foydalanuvchi mashinasini ko‘rib chiqamiz. 5.4-rasmda ko‘rsatilgan mikroprotsessorning registrlaridan tashqari, uning tarkibida 1 Mbayt hajmdagi xotiraning manzil kengligi va har biri 64 Kbayt hajmga ega bo‘lgan kiritish-chiqarishning ikkita portlar maydoni mavjud.

AX-DX deb belgilangan 16 razryadli umumiy vazifali registrlar bilan amallarni bajarishdan tashqari, bu registrlarning AL-DL va AN-DHlarning har bir baytiga murojaat qilishga ruxsat beriladi.

16 razryadli VR, SI, DI registrlar xotiraning bajariluvchi manzillarini tashkil etish uchun foydalaniladi, shuningdek, SP — stek ko‘rsatkichi, IP — dasturiy hisoblagich (SchK), Flags — bayroqlar registri bo‘lib, uning formati 5.5-rasmda keltirilgan va ular quyidagilarni bildiradi:

- CF — katta razryaddan ko‘chish yoki qarz olish (zayom);
- PF — paritetlik (natijada birlarning juftligi);
- AF — qo‘shimcha ko‘chish (3-razryaddan);
- ZF — nolli natija;
- SF — manfiy natija (belgi);
- OF — arifmetik to‘lib ketish belgisi;
- DF — manzilning oshishi yoki kamayishi;
- IF — INT kirish bo‘yicha tashqi uzilishni qoplaydi (IF = 1 bo‘lganda uzilishga ruxsat);
- TF — mikroprotsessorning qadamlar bo‘yicha bajarish rejimini boshqaradi.

TF = 1 bo‘lganda, har bir buyruq bajarilgandan so‘ng 1-vektor bilan avtomatik tarzda uzilish shakllanadi.

Manzil kengligini taqsimlash. K1810VM86 (i8086) mikroprotsessorlarining manzil kengligi manzil/ma’lumotlar shinasi razryadligi hamda manzil bilan aniqlanadi. Uning hajmi 2^{20} bayt = = 1 Mbaytga teng. Ushbu manzil kengligidan mikroprotsessorga bir vaqtning o‘ziga faqat to‘rtta segmentga kirishga ruxsat beriladi. Ularning ikkitasi (DS va ES) ma’lumotlarni joylashtirish uchun mo‘ljallangan bo‘lsa, CS — kod segmenti (dasturni joylashtirish uchun) va SS — stek segmenti bo‘lib hisoblanadi.

Segmentlarning o‘lchami buyruqlarning mantiqiy manzili, ma’lumotlar va stek orqali aniqlanadi. Buyruqlarning va stekning

5.6-rasm. Fizik manzilni shakllantirish (i8086).

(yuqorisi) mantiqiy manzillari 16 razryadli IP va SS registrlarda mos ravishda saqlanadi. Ma'lumotlarning mantiqiy manzili esa tizimlar buyrug'ining ko'p sonli usullarning biri orqali buyruq asosida hisoblanadi va u ham 16 bitni tashkil etadi.

Shunday qilib, ushbu mikroprotsessor har bir segmentining o'lchami 2^{16} bayt = 64 Kbaytni tashkil etadi. Manzilli kenglikdagi (uning boshlang'ich manzili) segmentning holati bir xil nomlangan segmentli registrning tarkibi bilan aniqlanadi.

Fizik manzilni shakllantirish jarayoni 5.6-rasmda keltirilgan. Ko'rinib turibdiki, manzilli maydonda segmentning chegarasi ixtiyoriy emas, balki segmentning boshlang'ich manzili 16 ga karrali bo'lgan holdagina o'rnatiladi.

Fizik manzilni shakllantirish uchun segmentli registrlar maxsus buyruq bo'lmaganda mustaqil tarzda quyidagicha tanlanadi: IP — manzil bo'yicha buyruq o'qilganda CS qo'llaniladi, ma'lumotlarga murojaat etilganda esa — DS yoki ES va nihoyat stekka murojaat etilganda SS qo'llaniladi. Buyruqlarga maxsus prefikslar (buyruqning oldiga maxsus belgi) qo'yish yordamida foydalanish uchun ixtiyoriy segmentli registrni belgilash mumkin (modifikatsiyaga tobe bo'lmagan CS:IP juftligidan tashqari). Segmentlarning chegaralari shunday tanlanishi mumkinki, segmentlar bir-biridan ajratilgan, bir-biri bilan kesishgan yoki to'liq mos tushgan bo'ladi.

Masalan, CS = SS = DS = ES = 0 yuklangan taqdirda, barcha segmentlar bir-biri bilan mos tushadi va nolinchi manzildan boshlanadi.

Buyruqlar tizimi. K1810VM86 (i8086) mikroprotsessorlari buyruqlar formati va manzillash usulining turli xilligi bilan farq qiladi. Buyruq uzunligi 1 dan to 6 gacha baytni tashkil etishi mumkin. Shundan boshlang'ich ikki bayt (ayrim hollarda birinchi bayt) amalning kodini, operandalarning soni va uzunligini va ularni manzillash usulini belgilaydi. Buyruqning qolgan baytlarida bevosita operand, to'g'ri manzil yoki siljish joylashishi mumkin.

Buyruqlarning ko‘pchiligi ikki manzilli hisoblanadi. Ulardan biri protsessor registrini va ikkinchisi xotira yacheykasini yoki registrni aniqlaydi.

Operand xotirada to‘g‘ridan to‘g‘ri yoki bilvosita bazali (VR, VX) registrlardagi yoki indeksli (SI, DI) registrlardagi kodlar, shuningdek, ularning yig‘indisi bilan manzillashtirilishi mumkin.

Nisbiy manzillashning ko‘plab variantlari nazarda tutilgan bo‘lib, ularda mantiqiy manzil ikkita yoki uchta qo‘shiluvchilar asosida hosil bo‘ladi. Mantiqiy manzilni hosil qiluvchi qo‘shiluvchilar protsessorning bitta yoki ikkita registrida va buyruqda joylashadigan 8 yoki 16 razryadli siljishda joy oladi.

Manzillash rejimlari yuqori darajadagi algoritmik tillardan samarali foydalanishni hisobga olgan holda loyihalangan. Masalan, oddiy o‘zgaruvchiga to‘g‘ri manzillash orqali murojaat qilish mumkin. Massiv elementiga esa VX, SI lar orqali bilvosita murojaat qilinadi. VR orqali manzillash rejimi stek segmenti ma’lumotlariga ruxsat olish uchun xizmat qiladi. Bunday usul rekursiv protseduralarni va yuqori darajali algoritmik tillar kompilatorlarini joriy etishda nihoyatda qulaydir.

Buyruqlar tizimi 113 ta bazali buyruqlardan tashkil topgan va ular quyidagi guruhlarga bo‘linadi:

◊ **ma’lumotlarni uzatish buyruqlari.** Ushbu buyruqlar quyidagi vazifalarni bajarishga xizmat qiladi:

- registr va xotira orasida axborot almashish;
- kiritish, chiqarish, jadval o‘zgartirishlarni amalga oshirish;
- umumiy vazifali registrga bajariladigan manzilni yuklash, 4 baytlik manzilli obyektni ko‘rsatgich-registrlarga yuklash (segmentning boshlang‘ich manzili va segmentdagi siljish);
- F bayroqlar registridagi ma’lumotlarni xotiraga, shuningdek, stekka va stekdan uzatish;

◊ **arifmetik buyruqlar:**

- belgili va belgisiz ikkilik sonlarni qo‘shish, ayirish, ko‘paytirish va bo‘lish;
- jamlangan ikkilik-o‘nlik sonlarni qo‘shish va ayirishni o‘nlik korreksiyalash;
- jamlanmagan ikkilik-o‘nlik sonlarni qo‘shish, ayirish, ko‘paytirish va bo‘lishda o‘nlik korreksiyalash;

◊ **mantiqiy buyruqlar va siljishlar:**

- inversiya, konyunksiya, dizyunksiya, tengsizlik amallari;
- TEST natijalarni kiritmagan holda bayroqlarni o‘rgatish bilan operandalar ustida razryadlar bo‘yicha konyunksiya amalini bajarish;

• bir yoki berilgan songacha siljitish (siljish konstantasi CLda joylashadi);

◊ **boshqarishni uzatish buyruqlari:** o‘tishlar, chaqirishlar, qaytishlar ikki xil ko‘rinishda bo‘ladi. Ularga ichki segmentdagi («yaqindagilar») va segmentlararo («uzoqdagilar») ko‘rinishlar kiradi. Yaqin uzatishlarda faqat IP yuklanadi, uzoq uzatishlarda esa IP va CS yuklanadi.

Boshqarishni uzatish to‘g‘ri (maqsadli manzil buyruqda bo‘ladi) yoki bilvosita (maqsadli manzil manzillashning standart rejimlaridan foydalangan holda hisoblanadi) bo‘lishi mumkin. Shartli o‘tishning 16 ta buyrug‘ida belgili va belgisiz sonlarning munosabatlari tekshiriladi. Siklni boshqarishning 4 ta buyrug‘i mavjud bo‘lib, ular SX registrda siklni takrorlash sonini joylashtirishga mo‘ljallangan;

◊ **ma’lumotlar zanjirini qayta ishlash buyruqlari** xotirada baytlar yoki so‘zlar ketma-ketligi ustida turli amallarni bajaradi. Ushbu buyruqlar bilan ma’lumotlar zanjirini qayta ishlash vaqti tegishli dasturlar bilan uni amalga oshirishdan ko‘ra ancha kamdir.

Nazorat savollari va topshiriqlari

1. Mikroprotsessorning dasturiy modeliga nimalar kiradi?
2. Siljish deganda nimani tushunasiz?
3. Mikroprotsessorning buyruqlar tizimiga nimalar kiradi?

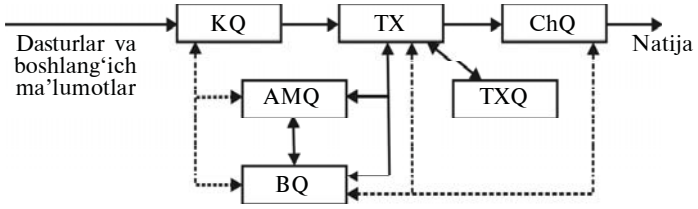
5.6. MARKAZIY QURILMA ARXITEKTURASINING ODDIY TURLARI

Bir turdagi markaziy qurilma va ma’lumotlarni saqlash qurilmasi har xil turdagi mashinalarda qo‘llanilishi mumkin. Ma’lumki, o‘z faoliyatini boshqarish mashinalarini ishlab chiqishdan boshlagan firmalar, o‘z mahsulotlarini takomillashtirib, tashqi qurilma konfiguratsiyasidan kelib chiqqan holda universal hamda boshqarish vazifalarini bajara oladigan mashinalar ishlab chiqarmoqdalar (Hewlett-Packard va Digital Equipment Corporation).

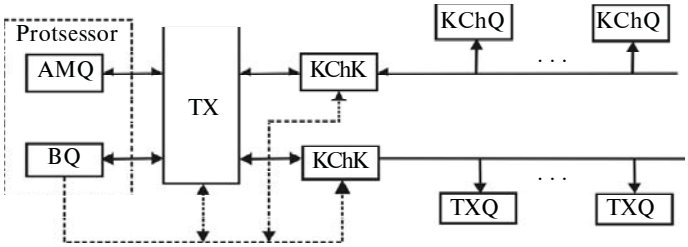
Ularni batafsil tahlil qilmagan holda, shuni aytish joizki, arxitekturaning asosiy klassik turlariga hisoblash mashinalarining yulduzsimon, shajaraviy (iyerarxik) va magistralli turlari kiradi.

Yulduzsimon arxitektura. Bunday arxitekturada markaziy qurilma (MQ) bevosita tashqi qurilma bilan bog‘langan bo‘lib, ularning ishini boshqaradi (oldingi mashinalar modeli).

Bunday klassik arxitekturaga ko‘ra (fon Neyman prinsipi asosida), bitta arifmetik-mantiqiy qurilmadan (AMQ) ma’lumotlar



5.7-rasm. Hisoblash mashinasining yulduzsimon arxitekturasi.



5.8-rasm. Hisoblash mashinasining shajaraviy arxitekturasi.

oqimi va bitta boshqarish qurilmasidan (BQ) buyruqlar (dasturlar) oqimi o'tadi. Bu bir protsessorli kompyuterdir.

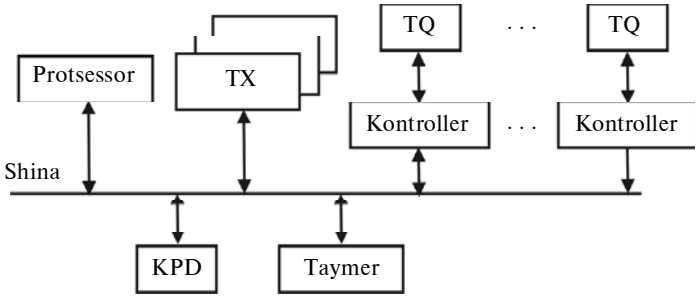
Hisoblash mashinasi beshta bazali komponentlardan iborat bo'lib, ular quyidagi qurilmalar turidan tashkil topgan (5.7-rasm):

- markaziy protsessor (MP), AMQ va BQdan iborat;
- xotira qurilmasi — xotira, shu jumladan, tezkor (TX) va tashqi xotiradan (TXQ) tashkil topgan;
- axborotlarni kiritish (KQ) va chiqarish (CHQ) qurilmalari − tashqi (periferiya) qurilmalari.

Shajaraviy arxitektura (5.8-rasm). Ushbu arxitekturaga ko'ra markaziy qurilma (MQ) periferiya protsessorlari (yordamchi protsessorlar, kanallar va h.k.) bilan bog'langan bo'lib, ularga tashqi qurilmalar guruhi ulanadi (IBM 360−375 tizimlari).

Magistralli struktura (umumiy shina — *unibas*, 5.9-rasm). Protsessor (protsessorlar) va xotira bloklari (TX) o'zaro va tashqi qurilmalar (tashqi qurilmalar kontrollerlari) bilan ushbu qurilmalar uchun umumiy bo'lgan ichki kanal yordamida o'zaro aloqa o'rnatadi (DEC, SHEHM, IBM PC-mashinalari).

Zamonaviy shaxsiy kompyuterlar magistralli arxitektura turiga mansub bo'lib, ularning funksional bloklari tizimli magistral deb nomlangan umumiy shina orqali bog'langan.



5.9-rasm. Hisoblash mashinasining magistralli arxitekturasi.

Fizik magistral ko‘p simli liniyalardan iborat bo‘lib, ular elektron sxemalarni maxsus portlarga ulash uchun xizmat qiladi.

Magistral simlarining majmuasi alohida guruhlarga ajratilgan va ular, o‘z navbatida, manzillar shinasi, ma’lumotlar shinasi va boshqarish shinasi deb nomlanadi.

Tashqi (periferiya) qurilmalar (printer, monitor) kompyuterga maxsus kontrollerlar (periferiya qurilmalarini boshqarish qurilmalari) orqali ulanadi. Kontroller maxsus qurilma bo‘lib, tashqi qurilmalarni yoki aloqa kanallarini markaziy protsessor bilan bog‘laydi va protsessorni o‘sha qurilmaning funksional ishlashini bevosita boshqarish vazifasidan ozod qiladi.

Ko‘rib chiqilgan EHMlarning barcha arxitektura elementlari quyidagi sxemali elementlar va bazali uzellarga asoslanadi:

- xotira – odatda, triggerlarning imkoniyatlari va xususiyatlaridan foydalaniladi;
- hisoblagich (registr) – buyruqlar manzilini shakllantirishga xizmat qiladi;
- summator (to‘la summator yoki yarim summator) – arifmetik-mantiqiy qurilmaning yadrosini tashkil etadi;
- deshifrator (masalan, buyruqlar deshifratori) – kodlarni deshifratsiyalash vazifasini bajaradi.

Nazorat savollari va topshiriqlari

1. EHMlarni qurishning qanday arxitekturalarini bilasiz va ular bir-biridan nima bilan farq qiladi?
2. Yulduzsimon va shajaraviy arxitekturalar biri-biridan nima bilan farq qiladi?
3. Zamonaviy mikroprotsessorlar qaysi arxitektura asosida qurilgan?

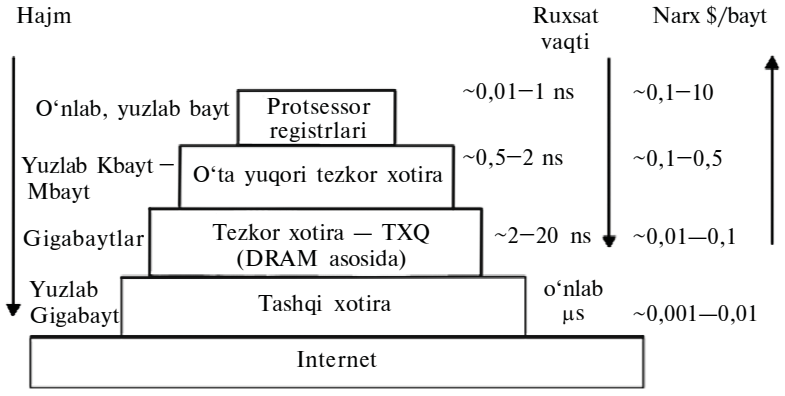
6-bob. HISOBLASH TIZIMLARIDA XOTIRANI TASHKIL ETISH STRUKTURASI

6.1. XOTIRANING SHAJARAVIY STRUKTURASI

Ma'lumki, EHMlarning samarali ishlashi ko'p jihatdan uning xotirasi xarakteristikalari bilan aniqlanadi. Hatto mikroprotsessorning ma'lumotlarni qayta ishlash yuqori tezligi hamda ularni uzatish tezligi ham bevosita xotiraga bog'liq. Shuning uchun azaldan xotiraga uchta asosiy shart qo'yib kelinadi: katta hajm, yuqori tezlik va past narx. Lekin, ushbu talablar bir-biriga o'zaro qarama-qarshi hisoblanadi. Chunki katta hajm bilan yuqori tezlikni yoki yuqori tezlikni past narx bilan ta'minlab bo'lmaydi. Shuning uchun zamonaviy EHMlarni ushbu talablarga javob beradigan va shajaraviy (iyerarxik) strukturaga ega bo'lgan turli tipdagi kompleks xotira qurilmalari tashkil etadi.

Demak, EHM xotirasini shajaraviy eslab qoluvchi qurilmalar tashkil etar ekan, bunday xotira qurilmalari bir-biridan ma'lumotlarga ruxsatning o'rtacha vaqti, hajmi va bir baytni saqlash narxi bilan farq qiladi (6.1-rasm).

Rasmdan ko'rinib turibdiki, sxemada yuqoridan pastga qarab harakat qilganda registrli xotiraning diskka qadar ma'lumotlarga ruxsat vaqti bir necha nanosekunddan o'nlab mikrosekundga qadar oshadi. Bunday holda xotira hajmi oshadi (registrlar yaxshi



6.1-rasm. Xotira qurilmalarining shajarasi.

holda 128 bayt saqlaydi, tashqi xotiraning hajmi chegaralanmagan), lekin bir bayt ma'lumotni saqlash hisobidan qaraganda uning narxi kamayadi.

Zamonaviy kompyuterlar xotirasining shajarasi bir necha darajada quriladi. Unda yuqori daraja hajmi jihatdan kichik va katta tezlikka ega, ammo uning narxi bayt hisobidan pastki darajalarga nisbatan yuqori bo'ladi.

Xotiraning shajaraviy strukturasi tahlili quyidagi xulosalar qilishga asos bo'ladi:

- hajm qancha katta bo'lsa, murojaat sikli (vaqti) shuncha uzun bo'ladi;
- bitta darajadagi barcha ma'lumotlar albatta nisbatan past darajada ham yotadi, teskari hol mavjud emas;
- xotiraning shajarasiga lokal xususiyat xosdir.

Bir darajada joylashgan xotira qurilmalarining xususiyatlari ham bir xil bo'ladi. Xotiraning shajarasi ko'plab darajalardan tashkil topgan, lekin real vaqt birligida mikroprotsessor faqat ikki o'zaro yaqin joylashgan darajalar bilan ish ko'radi. Xotira qurilmasi mikroprotsessorga qanchalik yaqin va ko'p jihatdan axborot almashib tursa, uning ma'lumot uzatish va qabul qilish tezligi ham shunchalik yuqori bo'lishi kerak. Aks holda u EHMning tezligiga salbiy ta'sir ko'rsatadi.

Hozirgi kunda zamonaviy EHMlarda turli darajadagi xotira qurilmalari qo'llaniladi. Ular quyidagi belgilari bo'yicha klassifikatsiyalanadi:

- fizik tipi bo'yicha (elektron xotira, magnit tashuvchilar, optikaelektron, gologrAfik, kriogenli va h.k.);
- ruxsat berishni tashkil etish bo'yicha (ketma-ket ruxsat beruvchi xotira qurilmalari. Ularda axborotlar ketma-ket yoziladi yoki o'qiladi. Masalan, CD-ROM);
- to'g'ridan to'g'ri ruxsat berish bo'yicha (bunday xotira qurilmalarida axborotni o'qish yoki yozish jarayoni istalgan vaqtda va istalgan yacheykada amalga oshiriladi. Masalan, tezkor (asosiy) xotira, doimiy xotira, kesh-xotira va h.k.);
- siklik (takrorlanuvchi) ruxsat berish bo'yicha (bunday xotira qurilmalarida axborotni yozish yoki o'qish joyi doimiy takrorlanadi. Masalan, vinchester);
- axborotni qidirish bo'yicha (adresli va assotsiativ xotira qurilmalari. Adresli xotira qurilmalarida har bir yacheyka o'z raqamiga—adresga ega va ushbu adres bo'yicha unga murojaat qilinadi.

Assotsiativ xotira qurilmalarida esa yacheykani qidirish kodlashtirilgan belgi—so‘rov asosida amalga oshiriladi, ya’ni bu jarayon «mazmun» bo‘yicha belgilanadi).

Axborotni saqlash va uni yozish imkoniyati bo‘yicha quyidagi xotira qurilmalari mavjud:

- mashinaning butun ishlash jarayonida axborot o‘zgarmaydigan doimiy xotira qurilmalari;
- tezkor xotira qurilmalari (yozish, saqlash va o‘qish rejimlarida ishlaydigan va mikroprotsessor bilan bevosita bog‘liq bo‘lgan xotira turi);
- yuqori tezkor xotira qurilmalari (yuqori tezlikka ega bo‘lgan triggerlardan tashkil topgan registrli xotira).

Xotira qurilmalarining asosiy xarakteristikalariga hajm, yozish zichligi, tezlik (axborot birligini qidirishga sarflanadigan vaqt), o‘tkazuvchanlik qobiliyati (bir sekundda uzatiladigan ma’lumotlar miqdori) kiradi.

Nazorat savollari va topshiriqlari

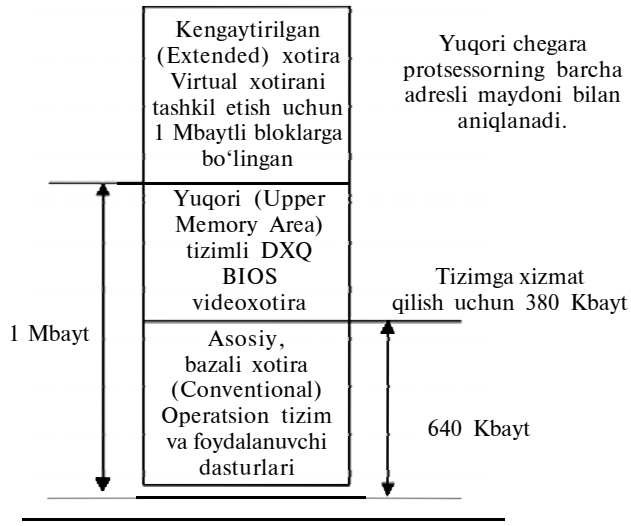
1. EHM xotiralariga qanday talablar qo‘yiladi?
2. Nima uchun EHM xotirasi shajaraviy strukturaga ega?
3. Xotira qurilmalari qanday belgilar bo‘yicha klassifikatsiyalanadi?

6.2. TIZIMLI XOTIRANI TAQSIMLASH

EHMlarning elektron xotirasi quyidagi ko‘rinishlarda namoyon bo‘ladi: tezkor xotira, kesh-xotira, doimiy xotira, yarim doimiy xotira, registrli xotira, buferli xotira, tashqi xotira.

Tezkor xotira. Tezkor (asosiy) xotira yoki tezkor xotira qurilmasi (Random Access Memory) o‘z tarkibida ko‘p murojaat qilinadigan va tezkor qayta ishlashi lozim bo‘lgan axborotlarni saqlash uchun xizmat qiladi. Tezkor xotira axborotlarni yozish, saqlash va uzatish rejimlarida ishlaydi. U mikroprotsessor bilan bevosita bog‘liq bo‘lganligi bois, uning tezligi (axborotni uzatishi, qabul qilishi) juda yuqori bo‘lishi talab etiladi. Tezkor xotiraning istalgan yacheykasidan ixtiyoriy tartibda ma’lumotlarni yozish yoki undan o‘qish imkoniyatlari mavjud.

Tezkor xotira qurilmasi bir necha sohaga bo‘lingan adresli maydonga ega. Bu sohalarning ayrimlaridan tizimning o‘zi foydalanadi, boshqa sohalari esa maxsus maqsadlarda qo‘llaniladi. 6.2-rasmda xotira adresli maydonining taqsimlanishi keltirilgan.



6.2-rasm. Xotira adresli maydonining taqsimlanishi (Intel—hamkor).

Yuqori xotira. (Upper Memory Area) — bu 384 Kbayt hajmdagi tizimli xotiraning yuqori chegarasida registratsiyalangan qismidir. Yuqori chegara bir necha qismlarga bo‘lingan:

- birinchi 128 Kbayt videoxotira sohasi bo‘lib, u videoadapterlar tomonidan foydalaniladi va ekranda namoyon bo‘lgan matn yoki grafika ko‘rinishlarini o‘zida saqlaydi;
- keyingi 128 Kbayt doimiy xotira mikrosxemasida yozilgan BIOS-adapterlar dasturlari uchun ajratilgan;
- oxirgi 128 Kbayt BIOS tizimli dastur uchun registratsiyalangan.

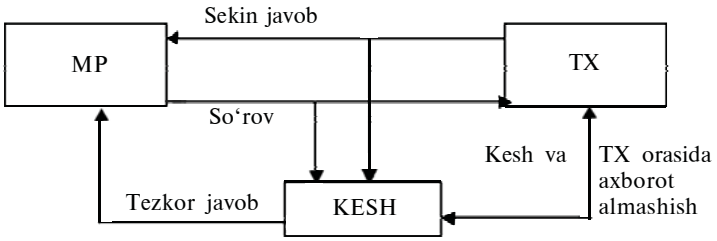
Videoadapter monitorga chiqariladigan grafik yoki simvolli axborotlarni saqlash uchun tizimli xotiraning bir qismidan foydalanadi.

Qo‘shimcha xotira. Yuqorida ta’kidlaganimizdek, zamonaviy EHMlarda tezkor xotiraning hajmi bir necha Mbaytni tashkil etadi. Masalan, Pentium-II bazasidagi tizimlarda tezkor xotiraning maksimal hajmi 64 Gbayt bo‘lsa, Pentium-III va Pentium-IV rusumli tizimlarda undan ham bir necha bor yuqoridir. Xotirani adresatsiyalashda birinchi megabaytdan tashqarida mikroprotsessor himoyalangan rejimda ishlashi kerak.

Hozirgi kunda xotira mikrosxemalarining ikki tipi keng tarqalgan: statik tezkor xotira qurilmasi — SRAM va dinamik tezkor xotira qurilmasi — DRAM.

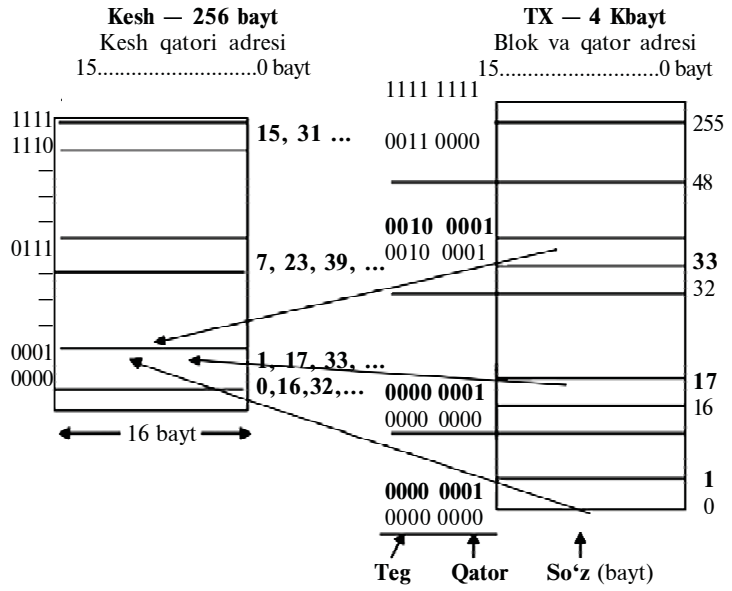
Kesh-xotira (cache memory). Kompyuter texnikasining asosiy masalalaridan biri — mikroprotsessor tezligiga mos bo‘lgan xotira tizimini qurish bo‘lib hisoblanadi. Bunday xotira tizimi operandlarni mikroprotsessor tezligi darajasida uzatishi talab etiladi. Ushbu muammoni kompyuter texnikasida kesh-xotirani qo‘llash orqali hal etish an’analari hozirgacha davom etmoqda. Kesh-xotira — tezkor xotira qurilmasi va mikroprotsessor (mikroprotsessorlar) orasida bufer vazifasini bajaradi. Birinchi darajali kesh-xotirani ko‘rib bo‘lmaydi, chunki u mikroprotsessor kristali ichida joylashgan. Shuning uchun kesh (ingl. — *cache*) — sirli, sirli joy ma’nolarini anglatadi.

Kesh-xotira o‘z tarkibida mikroprotsessor tomonidan oxirgi marta tezkor xotiradan qabul qilingan ma’lumotlar blokining nusxalarini saqlaydi. Agar ushbu ma’lumotlar mikroprotsessorga kerak bo‘lib qolsa, unda ularni kesh-xotiradan oladi va o‘zining takt chastotasi tezligida ularni qayta ishlaydi. Shu tariqa mikroprotsessorning tezkor xotiraga murojaatlari soni kamayishi hisobiga kompyuterning tezligi oshadi.



6.3-rasm. Kesh-xotiraning mikroprotsessor bilan aloqasi.

To‘g‘ridan to‘g‘ri akslantiruvchi kesh-xotira. Kesh-xotiraning eng oddiy tipi to‘g‘ridan to‘g‘ri akslantirish bo‘lib, unda har qanday qator tezkor xotiradan keshning bitta joyida paydo bo‘lishi mumkin. Masalan, kesh-xotira har biri 16 baytdan 16 ta qatorga ega bo‘lsin. Keshning har bir elementi (qatori) tezkor xotiradan roppa-rosa bitta qatorni sig‘diradi. Bunday holda biz tezkor xotiraning 4 Kbayt hajmini akslantiradigan 256 bayt hajmdagi kesh-xotiraga ega bo‘lamiz (6.4-rasm).



6.4-rasm. Tezkor xotira maydonini kesh-xotira maydonida akslantirish.

Rasmdan ko‘rinib turibdiki, tezkor xotirani kesh-xotirada bunday akslantirishda har bir blokka (hajm bo‘yicha) kesh-xotiraning bitta qatori ajratiladi.

Faraz qilaylik, mikroprotsessor 0010 0001 0110 adresga murojaat qilsin. Bunday holda biz keshning birinchi qatorini (0001) tekshirishimiz kerak. Agar unda xotiraning kerakli qatori joylashgan bo‘lsa, unda beshinchi baytni (0110) o‘qish kerak. Lekin keshning bu qatorida tezkor xotiraning 1, 17, 33 va h.k qatorlari ifodalangan bo‘lishi mumkin. Keshda aynan qaysi qator yozilganligini qanday bilish mumkin? Buning uchun TEGda (tag) ifodalangan axborot xizmat qiladi: bizning holda u to‘rt bitga (0110) teng, ya’ni bu faqat 33 qatordir (0010 0001). Shunday qilib, fizik adres bir nechta qismga bo‘linadi:

Teg	Qator	So‘z (bayt)

6.5-rasm. Ko‘rilgan hol uchun kesh-xotiraning strukturasi.

Bizning holda kesh-xotiraning strukturasi 6.6-rasmdagi ko‘rinishga ega bo‘ladi.

Qator adresi	Ishonchlilik bloki	Teg	Ma’lumotlar	Ushbu element foydalanadigan adreslar
1111				15, 31, 47, ...
—				
—				
—				
0001				1, 17, 33, . . .
0000				0, 16, 32, . . .

6.6-rasm. To‘g‘ridan to‘g‘ri akslantiruvchi kesh-xotira.

Rasmdan ko‘rinib turibdiki, tezkor xotiradan kesh-xotiraning birinchi qatoriga faqat birinchi qatorini, 17-qatorini, 33-qatorini va h.k. joylashtirish mumkin.

Kesh xotiraning har bir elementi quyidagi to‘rtta qismdan tashkil topgan:

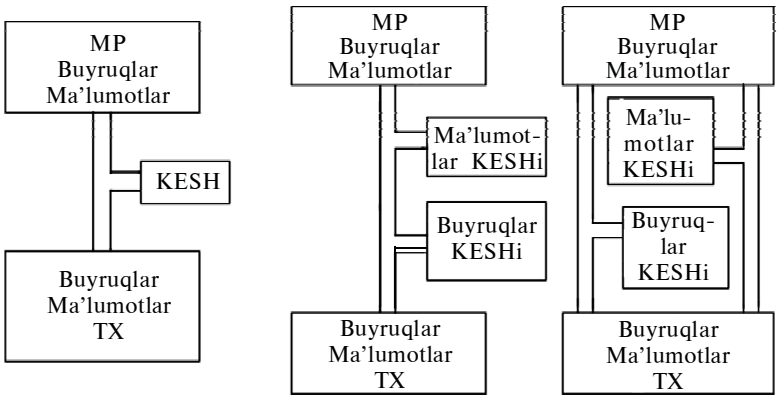
- kesh-xotiraning adresi;
- haqqoniylik bloki, boshqaruvchi axborot (elementda haqqoniylik ma’lumotlari borligi yoki yo‘qligini bildiradi);
- «teg» maydoni, ma’lumotlar kelib tushgan tegishli xotira satrini ifodalaydi;
- «ma’lumotlar» maydoni, asosiy xotira ma’lumotlarining nusxalarini saqlaydi, ma’lumotlar maydoni bitta qatorga 16 baytni joylashtiradi.

Ko‘rinib turibdiki, har bir yozuv o‘zida tezkor xotirada mavjud ma’lumotlar elementining adresini, ma’lumotlarning o‘zini, shuningdek, kesh-xotirada ma’lumotlar almashtirish algoritmini amalga oshiradigan qo‘shimcha boshqarish axborotini (modifikatsiya belgisi, ma’lumotlarga murojaatning chastotali belgisi) saqlaydi. Har bir qator uchun kesh-xotirada bitta boshqarish biti saqlanishi kerak. U ishonchlilik biti deyiladi. Tizim (kompyuter) elektr energiyasi bilan ta’minlanganda va ma’lumotlar vinchesterdan tezkor xotiraga yuklanganda, ishonchlilik bitlarining barchasi «0» holatga o‘rnatiladi. Qachonki kesh qatori birinchi marta tezkor xotiradan yuklansa, uning ishonchlilik biti «1» holatga o‘rnatiladi. Agar tezkor xotira bloki keshni chetlab o‘tib boshqa manbadan yangilansa (masalan, vinchesterdan), unda tizim

yuklanadigan blok keshda joylashganligini tekshiradi. Mabodo yuklanadigan blok keshda joylashmagan bo‘lsa, unda kesh-xotirada eskirgan ma’lumotlar bo‘lmasligi uchun uning ishonchlilik biti «0» holatga o‘rnatiladi.

Kesh-xotirani qurishning turli ko‘rinishlari. Kesh-xotira samaradorligini oshirish uchun quyidagi masalalar ijodiy hal etilishi talab etiladi: buyruqlar va ma’lumotlar birgalikda umumiy kesh-xotirada joylashishi kerakmi? Tabiiyki, buyruqlar va ma’lumotlar saqlanadigan umumiy kesh-xotirani yaratish oson variant hisoblanadi. Unda buyruqlarni va ma’lumotlarni olish avtomatik tarzda teng kuchli bo‘ladi. Shunday bo‘lsa-da, hozirgi kunda buyruqlarni bir kesh-xotiraga va ma’lumotlarni boshqasiga saqlaydigan bo‘-lak kesh-xotirani qo‘llash an’anasi ustuvor hisoblanmoqda.

Kesh-xotirani tashkil etishning asosan uchta usuli mavjud (6.7-rasm):



6.7-rasm. Kesh-xotirani qurishning turli ko‘rinishlari.

Birinchi holda *(a)* kesh ham buyruqlarni, ham adreslarni saqlaydi, ikkinchi holda *(b)* keshlar taqsimlangan, lekin ma’lumotlar va buyruqlar uchun shina umumiy hisoblanadi (bir fazali Garvard arxitekturasi). Uchinchi hol *(d)* to‘la Garvard arxitekturasi, deb nomlanadi va unda nafaqat ma’lumotlar va buyruqlar ikkita alohida keshlarda saqlanadi, balki ular uchun alohida shinalar ham mavjud. Alohida taqsimlangan kesh-xotira ma’lumotlar va operandlarga parallel ruxsat berish imkonini beradi.

Shuningdek, dasturni bajarish vaqtida buyruqlar o‘zgarmasligi bois, buyruqli kesh-xotira tarkibini qayta tezkor xotiraga yozish ehtiyoji qolmaydi.

Intel firmasi tomonidan ishlab chiqilgan i486 mikroprotsessorlaridan keyingi Pentium rusumli mikroprotsessorlarda ma’lumotlar va buyruqlar uchun alohida bo‘lak kesh-xotira qo‘llaniladi. Ularning har biri 8—16 Kbayt hajmga ega bo‘lib, ma’lumotlarga va buyruqlarga mustaqil ravishda murojaat qilishni ta’minlaydi. Bundan tashqari, ma’lumotlar kesh-xotirasi ikki karrali daraja prinsipi asosida qurilgan bo‘lib, kesh-xotiraning bitta qatoriga tegishli ikkita so‘zni bir vaqtda o‘qishni ta’minlaydi. Kesh-xotirani qayta yuklash samaradorligini oshirish uchun mikroprotsessorda 64-razryadli tashqi ma’lumotlar shinasi qo‘llaniladi.

Umuman olganda, xotira EHMning muhim resursi hisoblanadi va uni multidasturda operatsion tizim tomonidan oqilona boshqarish talab etiladi. EHMning ishlash jarayonida tezkor xotiraning operatsion tizim egallamagan barcha qismlari taqsimlanadi. Odatda, operatsion tizim eng kichik adreslarda joylashadi, lekin u eng yuqori adreslarni ham egallashi mumkin.

Xotirani boshqarish bo‘yicha operatsion tizimning funksiyalariga quyidagilar kiradi:

- xotiraning band bo‘lgan va bo‘sh bo‘lgan sohalarini kuzatib borish;
- ma’lumotlarni qayta ishlash jarayoni uchun xotiradan joy ajratish;
- ma’lumotlarni qayta ishlash jarayoni tugagandan so‘ng xotirani bo‘shatish;
- asosiy xotiraning hajmi kamayib borganda hisoblash jarayonida ishlatilgan ma’lumotlarni yana tezkor xotiradan vinchesterga siqib chiqarish;
- tezkor xotirada bo‘sh joylar paydo bo‘lganda ma’lumotlarni vinchesterdan qayta xotiraga yuklash;
- dastur adreslarini fizik xotiraning aniq sohasiga sozlash.

Yuqorida ta’kidlab o‘tilgan barcha fikrlarni inobatga olgan holda ayrim xulosalarni keltiramiz.

Xotira — bu EHMning asosiy qurilmalaridan biri bo‘lib, u ma’lumotlarni va dasturlarni saqlash uchun xizmat qiladi.

EHM xotirasi — bu xotira qurilmalarining majmuasi bo‘lib, ular EHMni tashkil etuvchi qurilmalar tarkibiga kiradi.

Kompyuter xotirasi — bir nechta darajalardan tashkil topgan shajaraviy tizimdir.

EHMning ichki xotirasi turli-tuman fizik prinsiplar asosida qurilishi mumkin, lekin o‘zining funksional belgilanishi bo‘yicha tezkor xotiraga, doimiy xotiraga, qayta dasturlanadigan doimiy xotiraga va h.k. turlarga bo‘linadi.

Zamonaviy tezkor xotira yarimo‘tkazgichli mikrosxemalar asosida statik va dinamik tiplar bo‘yicha ishlab chiqariladi.

Mikroprotsessorda ma’lumotlarni qayta ishlash vaqti va tezkor xotiraga murojaatlar vaqtini moslashtirishni hal etish uchun ko‘p darajali bufer strukturasi vazifasini bajaruvchi kesh-xotira qo‘llaniladi. Zamonaviy kompyuterlarda kesh-xotira ikki va uch darajali sxema bo‘yicha ishlab chiqiladi. Birinchi darajali kesh bevosita mikroprotsessor kristalida joylashadi, ikkinchi va uchinchi darajali keshlar esa bevosita tizimli platada (ona platada) joylashtiriladi.

Yuqorida ta’kidlaganimizdek, xotira har qanday EHMning asosiy resursi hisoblanadi. Uni multipleksorli operatsion tizim tomonidan sinchkovlik va katta e’tibor bilan boshqarish talab etiladi.

Nazorat savollari va topshiriqlari

1. Xotiraning adresli maydonidan EHM qanday foydalanadi?
2. Dinamik va statik xotiralarning farqlarini izohlang.
3. Kesh xotiraning qanday afzallik tomonlari mavjud?
4. Xotirani boshqarish jarayonida operatsion tizim qanday funksiyalarni bajaradi?

6.3. XOTIRANING MANTIQIY TASHKIL ETILISHI

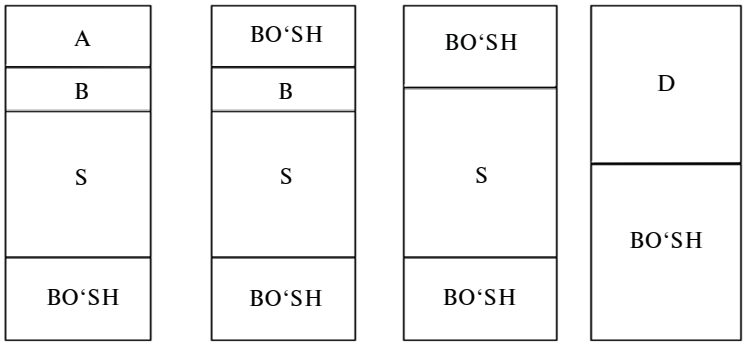
Hozirga qadar tezkor (asosiy) xotira kompyuterning eng muammoli resursi bo‘lib kelgan va bo‘lib kelmoqda. Shuning uchun barcha operatsion tizimlarning birinchi darajadagi funksiyasi xotirani raqobatbardosh foydalanuvchi jarayonlar orasida taqsimlashni ta’minlashdir. Odatda, multidasturli EHMlarning tezkor xotirasida doim operatsion tizimning yadrosi saqlanadi. Dasturlar operatsion tizimning yadrosidan EHMning ish jarayonida ko‘p marta bajariladi va ularni bajarish vaqti nisbatan ko‘p emas. Operatsion tizimning kerakli modullari tashqi xotiradan tezkor xotiraga yuklanadi va uning bir qismini egallaydi.

Tezkor xotiraning qolgan qismida bir nechta dasturlar, shuningdek, ular uchun kerak bo‘ladigan ma’lumotlar saqlanadi va ular multidastur rejimida bajariladi. Xotirani samarali taqsimlash deganda foydalanuvchilar va tizim vositalarining ehtiyojini qondirish nazarda tutiladi. Albatta, bu talablar ko‘p hollarda qarama-qarshidir. Tezkor xotira resurslarini taqsimlashning uchta strategiyasi mavjud bo‘lib, ularga statik va dinamik taqsimlanish hamda svoping kiradi.

Statik taqsimlashda tezkor xotiraning barcha zaruriy resurslari jarayonga ajratiladi. Bunda xotira zaruriy uzunlikdagi yagona blokka taqsimlanadi. Blokning boshlanishi bazali adres orqali aniqlanadi. Bajariladigan dastur blokning nisbatan boshidagi adreslarga yoziladi. Dasturni bajarish jarayonida buyruqning yoki operandning fizik adresi blokning bazali adresi va blokdagi nisbiy adres yig‘indisidan shakllanadi. Bazali adresning qiymati dasturni tezkor xotiraga yuklash paytida o‘rnatiladi.

Misol. Tezkor xotira 10 Mbayt hajmga ega bo‘lsin. A, B, C, D dasturlarni bajarish uchun esa, mos ravishda, quyidagi xotira hajmi talab etilsin: A—2 Mbayt, B—1 Mbayt, C—4 bayt, D—4 Mbayt. 6.8-rasmda xotiraning boshlang‘ich taqsimoti va ayrim dasturlar bajarilgandan so‘ng xotiraning taqsimoti keltirilgan.

Rasmdan ko‘rinib turibdiki, xotirani statik taqsimlashda D dastur tezkor xotiraga oldingi barcha dasturlar bajarilgandan so‘ng



6.8-rasm. Xotiraning statik taqsimlanishi:

a) boshlang‘ich taqsimot; *b)* birinchi dastur tugagandan keyingi taqsimot; *d)* ikkinchi dastur tugagandan keyingi taqsimot; *e)* keyingi S dastur tugagandan so‘nggi taqsimot.

yuklanishi mumkin, vaholanki uni bajarish uchun xotira hajmi A dastur bajarilgandan so‘ng ham yetarli bo‘lgan. Shu bilan birga, multidasturli EHMning ish ko‘rsatkichlarini yaxshilash uchun tezkor xotiraga bajarilishi kerak bo‘lgan ko‘p sonli dasturlarning bo‘lishi talab etiladi.

Tezkor xotirani svoping *(svapping)* usulida taqsimlashda operatsion tizim joriy jarayonni (dastur va ma’lumotlarni) butunlay tezkor xotiraga joylashtiradi, lekin zarurat tug‘ilganda ularni butunlay tezkor xotiradan olib tashlab, uni boshqa jarayon bilan almashtirishi mumkin.

Xotirani dinamik taqsimlashda har bir dasturga boshlang‘ich vaqtda unga kerakli xotira hajmining bir qismini ajratadi. Xotiraning qolgan qismi esa dastur uchun real ehtiyoj tug‘ilganda ajratiladi. Tizimli vositalar tezkor xotirada bo‘lmagan dasturning kerakli qismiga ehtiyoj tug‘ilishini doimiy ravishda kuzatib borishlari talab etiladi. Shundan so‘ng ushbu dasturga kerakli xotira blokini ajratib, tashqi xotiradan (vinchester) dasturning talab etilgan qismi joylashtiriladi. Buning uchun axborotning ayrim bloklarini oldindan tezkor xotiradan tashqi xotiraga siljitish talab etilishi mumkin. Bunday siljitishni foydalanuvchi sezmaydi va bu jarayon oz bo‘lsa-da, dastur bajarilishi tezligini sekinlashtiradi.

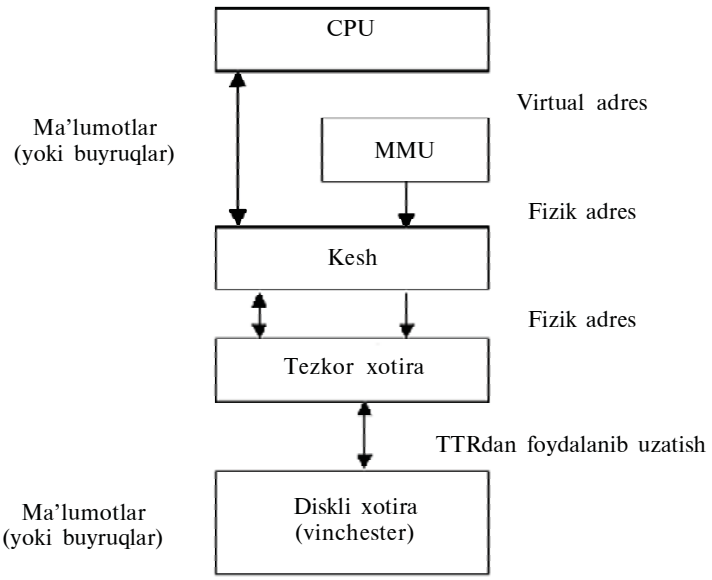
Xotirani dinamik taqsimlash virtual xotira tushunchasi bilan bog‘lanib ketgan bo‘lib, tashkiliy strukturasi bo‘yicha bir-biriga juda yaqindir.

Virtual xotira. Virtual xotira prinsipi bo‘yicha foydalanuvchi o‘z dasturini tayyorlashda fizik (real) tezkor xotira bilan emas, balki hajmi xotiraning butun adresli maydoniga teng bo‘lgan virtual (bo‘lishi mumkin bo‘lgan) bir darajali xotira bilan ish ko‘radi. Virtual xotira tezkor va tashqi xotira resurslaridan samarali foydalanish vositasi bo‘lib, u operatsion tizimning amaliy dasturlariga xotira maydonini ajratib beradi.

Virtual xotira oldingi avlod EHMlarida ham qo‘llanilgan va foydalanuvchilar dasturlarini bir-biridan himoya qilish va bir vaqtning o‘zida ishlayotgan dasturlar uchun tezkor xotirani samarali taqsimlash uchun xizmat qilgan.

Virtual xotira quyidagi masalalarni yechishga xizmat qiladi:

• ma’lumotlarni turli tipdagi xotira qurilmalariga joylashtiradi. Masalan, dasturning bir qismini tezkor xotiraga, yana bir qismini vinchesterga joylashtiradi;



6.9-rasm. Virtual xotirani tashkil etish
(TTR—xotiraga to‘g‘ridan to‘g‘ri ruxsat, MMU — xotirani boshqarish bloki).

• ma’lumotlarni zarurat darajasida turli tipdagi xotira qurilmalari ichida aralashtiradi;

• virtual adreslarni fizik (real) adreslarga o‘zgartiradi.

Bu amallarning barchasi dasturchining ishtirokisiz avtomatik ravishda bajariladi.

Virtual adreslarni fizik adreslarga o‘zgartirishni xotirani boshqarish moduli yoki xotira dispetcheri (Memory Management Unit) amalga oshiradi. Agar kerakli ma’lumotlar (yoki buyruqlar) tezkor xotirada mavjud bo‘lmasa, dispetcher ularni tashqi xotiradan tezkor xotiraga uzatadi.

Xotirani sahifali tashkil etish. Oldingi bo‘limlarda biz mantiqiy xotira adres maydonining fizik adresning real hajmiga mos kelmasligi xususida fikr yuritgan edik. Endi ushbu nomutanosiblikning zamonaviy mikroprotsessorlarda qanday bartaraf etilishini aniq misollar orqali ko‘rib chiqamiz.

Misol: Faraz qilaylik, bizning kompyuterimiz 16-bitli adres maydoniga va bor-yo‘g‘i 4096 so‘zli tezkor xotiraga ega bo‘lsin. Ushbu kompyuterda ishlaydigan dastur 65536 so‘zga (216 =65536) murojaat qilishi mumkin edi, lekin bunga so‘zning o‘zi yo‘qligi

ko‘rinib turibdi. Virtual xotirani ixtiro qilishdan oldin 4096-adresga teng yoki undan katta bo‘lgan barcha adreslar mavjud bo‘lmagan foydasiz hisoblanar edi.

Adres maydoni va xotira adreslari tushunchalarini alohida ajratish g‘oyasini quyidagicha izohlash mumkin. Istalgan vaqtda xotiraning 4096-so‘ziga to‘g‘ridan to‘g‘ri ruxsat olish mumkin, lekin bu aynan ular 0-dan 4096-gacha adreslarga mos ekanligini anglatmaydi.

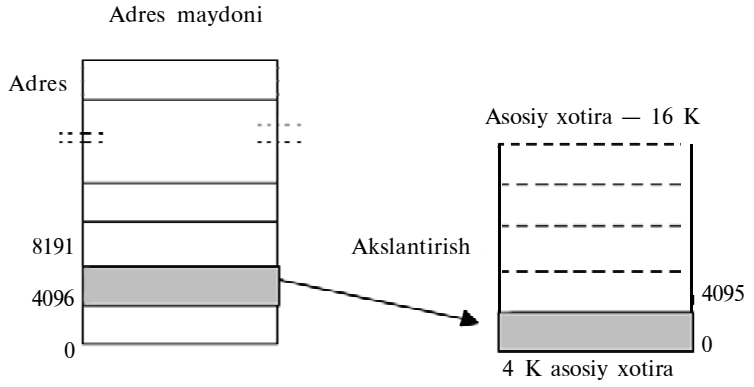
Masalan, biz kompyuterga 4096-adresga murojaat qilganda xotiradan 0-chi adresdagi so‘z foydalanilishi, 4097-adresdagidan 1-so‘z, 8191-adresga murojaat qilganda esa 4095-va h.k. adresdagi so‘z foydalanilishi kerakligi xususidagi xabarni berishimiz mumkin edi. Boshqacha qilib aytganda, mantiqiy adresli maydonni fizik xotiraning haqiqiy adreslarida akslantirdik (6.10-rasm).

Bu yerda haqli bir savol tug‘iladi: Agar dastur 8192-dan to 12287-adreslarning biriga o‘tganda nima sodir bo‘ladi? Virtual xotirasi bo‘lmagan mashinalarda «Xotiraning mavjud bo‘lmagan adresi» — jumlasi paydo bo‘ladi. Virtual xotiraga ega bo‘lgan mashinalarda esa quyidagi amallar ketma-ketligi bajariladi:

1. 4096-dan to 8191-gacha so‘zlar diskka (vinchesterga) yuklatiladi.

2. 8102-dan to 12287-gacha so‘zlar diskdan tezkor xotiraga yuklanadi.

3. Adreslarni akslantirish o‘zgaradi: endi 8192-dan to 12287-gacha bo‘lgan adreslar xotiraning 0-dan to 4095-gacha yacheykalariga mos bo‘ladi.



6.10-rasm. 4096-dan 8191-gacha xotiraning virtual adreslari asosiy xotiraning 0-dan to 4095-gacha adreslarida akslantiriladi.

4. Dasturni bajarish jarayoni davom etadi.

Endi, mabodo tezkor xotiraga 4 Kbaytdan yana uchta blok qo‘shilsa, unda biz 4 Kbayt fizik xotiraga 64 Kbaytli adresli maydonni to‘la akslantirish imkoniga ega bo‘lamiz.

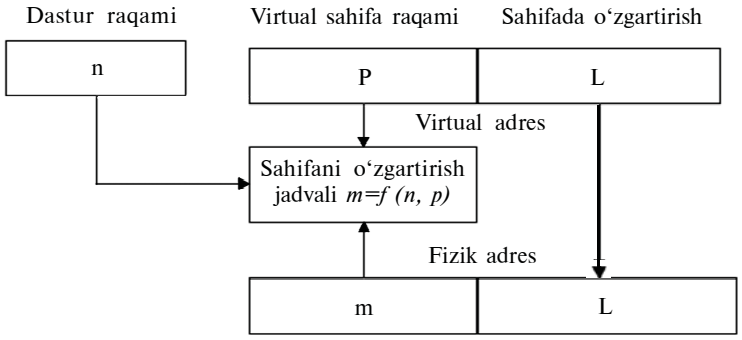
Avtomatik tarzda bunday joylashtirish texnologiyasi xotirani sahifali tashkil etish deyiladi. Diskdan o‘qiladigan dasturning qismlari esa sahifalar deyiladi.

Shuni ta’kidlash joizki, xotirani sahifali tashkil etish adresli maydon hajmiga ega bo‘lgan asosiy xotiraning mavjudligi xususidagi fikrga olib keladi. Dasturchi dasturni yozishda xotiraning sahifali tashkil etilganligini bilmasligi ham mumkin. Adreslarni sahifali translatsiyalash xotirani boshqarish bloki (MMU—Memory Management Unit) tomonidan bajariladi. MMU mikroprotsessorda joylashgan bo‘lib, kataloglardan va tezkor xotiradagi fizik strukturadan foydalanadi. MMU bloki chiziqli adresni fiksirlangan o‘lchamdagi (4 Kbayt, 4 Mbayt, 2 Mbayt) virtual sahifalarga bo‘ladi. Shunday sahifalarga fizik adresning adresli maydoni ham bo‘linadi.

Adreslarni o‘zgartirish. Sahifali virtual xotiradan oddiy va ko‘p hollarda har bir virtual xotira va asosiy fizik xotira bir xil o‘lchamdagi bloklardan yoki sahifalardan tuzilgan, deb tasvirlanadi. Qulaylik uchun sahifaning hajmi doim ikkiga karrali songa teng bo‘lganda tanlanadi. Bunday holda, agar virtual adresning umumiy uzunligi N bo‘lganda (qaysidir 2, 16, 32, 64-daraja), unda u ikkita maydondan tashkil topgan struktura sifatida qaraladi. Birinchi maydon (N-M+1) adres razryadini egallaydi va virtual xotira sahifasining raqamini beradi. Ikkinchi maydon (M-1) razryadni egallaydi va xotiraning adreslanadigan elementigacha sahifaning ichidagi o‘zgartirishni belgilaydi. Virtual adresning apparatli interpretatsiyasi 6.11-rasmda keltirilgan.

Ushbu rasm konseptual darajada ifodalangan. Unda sahifalar jadvali qayerda saqlanishi ko‘rsatilmagan bo‘lsa-da, virtual xotira xususida tasavvur hosil qilishga imkon beradi. Shuni ta’kidlash joizki, sahifali tashkil etilgan virtual xotirali zamonaviy kompyuterlarda sahifalarning barcha jadvallari tezkor xotirada saqlanadi.

Xotirani segmentli tashkil etish. Xotirani sahifali taqsimlashni tashkil etish bir qator kamchiliklarga ega. Xotirani bunday tashkil etishda virtual adreslar qaysidir maksimal adresga qadar birinketin keladi. Uni bir o‘lchovli xotira deb aytish mumkin. Ko‘plab sabablarga ko‘ra bir nechta alohida virtual adresli maydonlarni tashkil etish qulaydir.



6.11-rasm. Virtual adresni fizik adresga o‘zgartirish.

Darhaqiqat, kompilator kompilatsiya jarayonida bir nechta yaratilgan jadvallarga ega bo‘lishi mumkin. Jumladan:

- o‘zgaruvchilarning ismi va atributlariga ega bo‘lgan simvollar jadvali;
- boshlang‘ich kod;
- konstantalar jadvali;
- protseduralarni chaqirish jadvali;
- xizmat qiladigan stek va h.k.

Kompilatsiya jarayonida ayrim jadvallar kengayishi, boshqalari esa kamayishi mumkin. Stek esa oldindan bashorat qilib bo‘lmaydigan holga tushadi. Bir o‘lchovli maydonda bu jadvallardan hamkorlikda foydalanish ayrim muammolar tug‘diradi. Masalan, ularning ayrimlari to‘lib ketadi, vaholanki, boshqalari yarim bo‘sh holda yotadi. Bunday vaziyatda nima qilish kerak va dasturchini jadval hajmini boshqarish tashvishidan qanday qilib ozod etish kerak degan savol tug‘iladi.

Buning uchun chegaralari, hajmlari va ularning qo‘shilmalari «suzadigan» ko‘plab mustaqil adres maydonlarini yaratish zarur. Bunday bloklar segmentlar deyiladi.

Xotira ixtiyoriy uzunlikdagi bitta yoki ko‘plab bloklardan, segmentlardan mantiqan tashkil etilishi mumkin. Himoyalangan rejimda mantiqiy xotiraning hajmini 5 Mbaytdan bo‘lgan sahifalarga bo‘lish mumkin, ularning har biri fizik xotiraning ixtiyoriy joyida akslantirilishi mumkin. Adreslarni segmentatsiyalash va sahifani translatsiyalash birgalikda yoki alohida amalga oshirilishi mumkin. Segmentatsiya mantiqiy xotirani amaliy darajada tashkil etish vositasi bo‘lib hisoblansa, adreslarni sahifali translatsiyalash esa tizimli darajada tashkil etishni belgilaydi.

Svoping. Virtual xotirani tashkil etishning yana bir turi svoping bo‘lib hisoblanadi. Svoping metodi bo‘yicha ayrim jarayonlar (odatda, kutish holatida bo‘lganlar) vaqtinchalik diskka yuklanadi. Operatsion tizim ularni o‘z nazoratida qoldiradi va jarayonni faollashtirish sharti kelganda diskning svoping sohasida joylashgan bu jarayon tezkor xotiraga joylashadi.

Agar tezkor xotirada bo‘sh joy yetarli bo‘lmasa, u holda boshqa jarayon yuklanadi. Virtual xotirani svoping usulida tashkil etish oldin ko‘rib chiqilgan usullardan farq qiladi. Svoping usulida jarayon diskdan tezkor xotiraga yaxlit ko‘chiriladi va bunday paytda jarayon tezkor xotirada umuman bo‘lmasligi mumkin. Yuqorida bayon etilgan barcha fikrlar asosida quyidagi xulosalarni qilish mumkin:

Xotirani mantiqiy tashkil etish, xotiraning adres maydoni va uning real fizik hajmi orasidagi nomutanosiblikni bartaraf etish uchun zarurdir.

Xotirani stekli tashkil etish qism dasturlarni chaqirishda, ma’lumotlarni vaqtinchalik saqlashda qo‘llaniladi.

Virtual xotirani tashkil etish masalalari xotirani mantiqiy tashkil etish orqali amalga oshiriladi.

Xotirani mantiqiy tashkil etishda stekli, segmentli, svoping va boshqa turlardan foydalaniladi.

Nazorat savollari va topshiriqlari

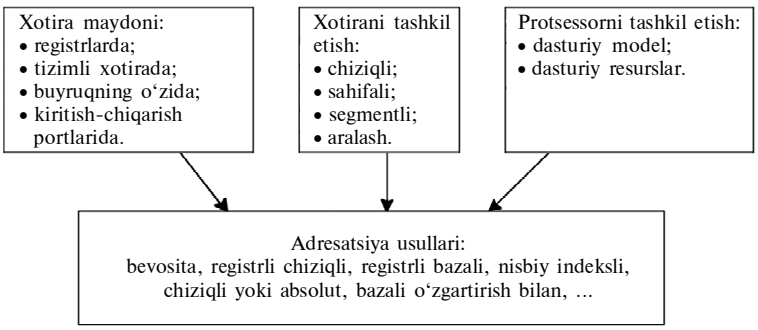
1. Xotirani statik va dinamik taqsimlash qanday amalga oshiriladi?
2. Virtual xotira qanday afzalliklarga ega va u qanday vazifalarni bajarishga xizmat qiladi?
3. Virtual adresni fizik adresga o‘zgartirish jarayonini tushuntiring.
4. Nima uchun xotirani mantiqiy tashkil etish ehtiyoji tug‘ilgan?

6.4. ADRESATSIYA USULLARI

Adresatsiya usullari va rejimlari uchta asosiy faktorlarga bog‘liq:

- operandlarni saqlash uchun xotira maydonini tashkil etish: ichki registrlarda, tizimli xotirada, stekli xotirada, kiritish-chiqarish portlarida;
- xotirani strukturali tashkil etish: chiziqli, sahifali, segmentli, aralash;
- mikroprotsessorning dasturli modeli, dasturiy resurslar.

Xotira maydoni buyruqlarning kodini va ma’lumotlarni saqlash uchun belgilangan. Unga ruxsat berishning bir qator



6.12-rasm. Adresatsiya usullari va rejimlari.

adresatsiya usullari (24 dan ortiq) mavjud. Operandlar mikroprotsessorning ichki registrlarida joylashishi mumkin (qulay va tezkor variant). Ular tizimli xotirada ham saqlanishi mumkin (eng keng tarqalgan variant). Va nihoyat, operandlar buyruqning o‘zida va kiritish-chiqarish qurilmalarida bo‘lishi mumkin (kam uchraydigan variant).

Operandlarning joylashish o‘rnini aniqlash buyruq kodi orqali amalga oshiriladi. Kirish operandini qayerdan olish kerak va chiqish operandini qayerga yozish kerakligi xususida ko‘plab usullar mavjud bo‘lib, ulardan buyruq kodi keng foydalaniladi. Bu usullar adresatsiya usullari deyiladi. Adresatsiya usullarini samarali tanlash ko‘p jihatdan mikroprotsessorning samarali ishlashini belgilaydi.

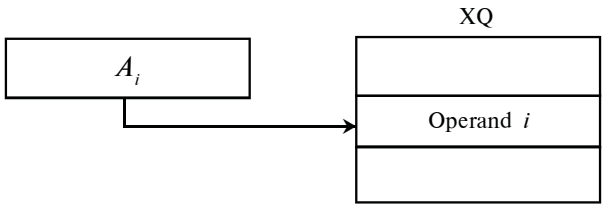
Registrli xotirada va tezkor xotirada joylashgan operandlarning adresatsiya mexanizmi bir-biridan farq qiladi. Registrli xotira uchun faqat to‘g‘ri adresatsiyaga ruxsat beriladi. Ushbu holda buyruqda operand saqlangan registrning raqami ko‘rsatiladi. 16-razryadli operand quyidagi registrlarda saqlanishi mumkin: AX, BX, CX, DX, DI, SI, SP, BP. 8-razryadli operand esa quyidagi registrlarda saqlanadi: AL, AH, BL, BH, CL, CH, DL, DH.

Tezkor xotiraning adresatsiya tizimi o‘zining ayrim xususiyatlariga ega bo‘lib, bunday xususiyatlar tezkor xotiraning segmentlarga bo‘lish bilan va segmentning boshlang‘ich adresini ko‘rsatish bilan bog‘liq bo‘lgan segmentli registrlar guruhi bilan bog‘liqdir.

To‘g‘ri yoki absolut adresatsiya. Operandning fizik adresi buyruqning adresli qismida joylashadi. Formal belgilanishi:

$$\text{Operand } i = (A_i),$$

bu yerda, A_i — buyruqning i adresli maydonida joylashgan kod.



6.13-rasm. To‘g‘ri adresatsiya.

Masalan, mov Al,[2000] — ushbu buyruq 2000h adresida joylashgan operandni AL registrga uzatadi.

Add R1,[1000] — R1 registrida joylashgan operandni xotiraning 1000h adresidagi yacheykaga joylashgan operand bilan qo‘shish va natijani R1 ga yozish.

To‘g‘ri adresatsiyani asosiy xotiraga hamda registrli xotiraga murojaat qilishda foydalanishga ruxsat beriladi.

Bevositali adresatsiya. Buyruqda operandning adresi emas, balki bevosita operandning o‘zi saqlanadi:

$$\text{Operand } i = A_i$$

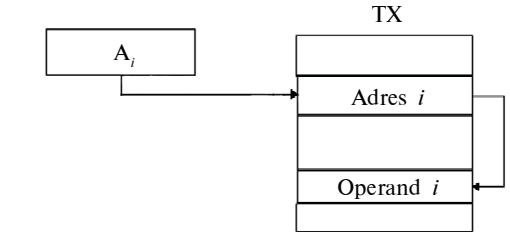
6.14-rasm. Bevosita adresatsiya.

Bevosita adresatsiya amallarni bajarish tezligini oshirish imkonini beradi. Chunki butun buyruq (operand bilan birga) xotiradan bir vaqtning o‘zida o‘qiladi va buyruqni bajarish vaqtida mikroprotsessorning maxsus buyruq registrida (BR) saqlanadi. Ammo bevosita adresatsiyadan foydalanishda buyruqlar kodining ma’lumotlarga bog‘liqligi paydo bo‘ladi va bu jarayon operandning har bir o‘zgarishida dasturni o‘zgartirishni talab etadi.

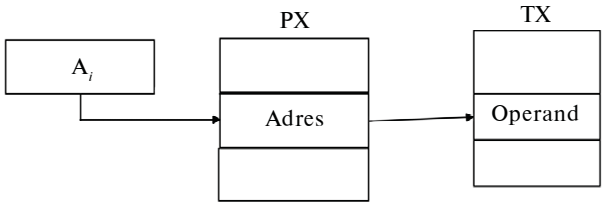
Masalan, mov eax,0f0f0f0f — konstanta 0f0f0f0f h ni eax registrga yuklash.

Bazali adresatsiya. Ushbu usulda operand adresi joylashgan xotira yacheykasi adresini buyruqning adresli qismi ko‘rsatadi:

$$\text{Operand } i = ((A_i)).$$



a) TX yacheykasi orqali



b) Registrli xotira orqali

6.15-rasm. Bazali adresatsiya.

Operand adresni registrli xotirada saqlashda — tezkor xotirada bazali adresatsiyani qo‘llash orqali biz adres maydoni uzunligini qisqartirishimiz mumkin. Bazali adresatsiya registri xotirada joylashgan operandlarga nisbatan qo‘llanilmaydi.

Misol: mov eax,[eci+4] — xotira yacheykasidagi operandni ECI (indeks registri) registrga plus 4 siljish bilan uzatish. Uning adresi EAX registrida joylashgan.

Registrli adresatsiya. Bu usulda operand mikroprotsessorning ichki registrida joylashadi.

Masalan: mov eax, cro — EAXga CRO (boshqarish registri) ma’lumotlarini uzatish.

Indeksli adresatsiya (almashtirish bilan). Umumiy belgili registrdagi ma’lumotlar samarali adres komponentasi sifatida qo‘llaniladi.

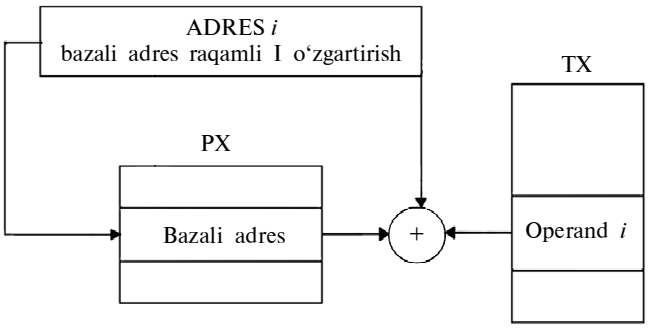
Masalan: sub array [esi], 2 — ESI registr ko‘rsatgan massiv elementidan ikkini ayirish.

Nisbiy adresatsiya. Ushbu usul xotira mantiqan bloklarga bo‘linganda va segment deb nomlangan paytda qo‘llaniladi. Bunday holda xotira yacheykasi adresi ikkita tashkil etuvchidan iborat bo‘ladi: segmentning boshlang‘ich adresi (bazali adres) va segmentda operandning almashtirilgan adresi.

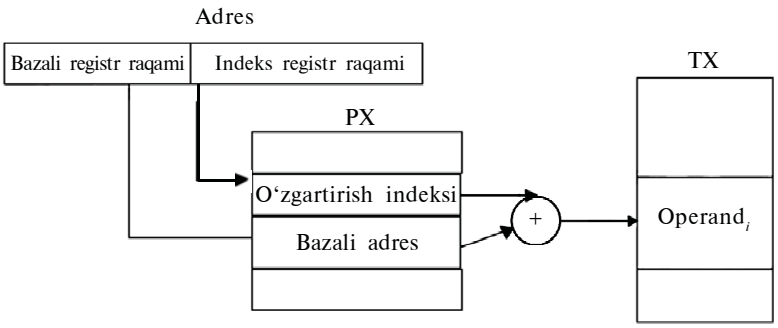
Operand adresi yig‘indidan aniqlanadi. Yig‘indi bazali adres va bu bazaga nisbatan amalga oshirilgan o‘zgartirishlarning qo‘shilishi natijasida hosil bo‘ladi:

$$\text{Operand}_i = (\text{baza}_i + \text{o‘zgartirish}_i).$$

Bazali adres va o‘zgartirishni berish uchun oldin ko‘rilgan adresatsiya usullari qo‘llanilishi mumkin. Odatda, bazali adres registrli xotiraning biror registrida joy oladi, o‘zgartirish esa buyruqning o‘zida yoki registrda berilishi mumkin.



6.16-rasm. Nisbiy adresatsiya.



6.17-rasm. Bazali-indeksli adresatsiya.

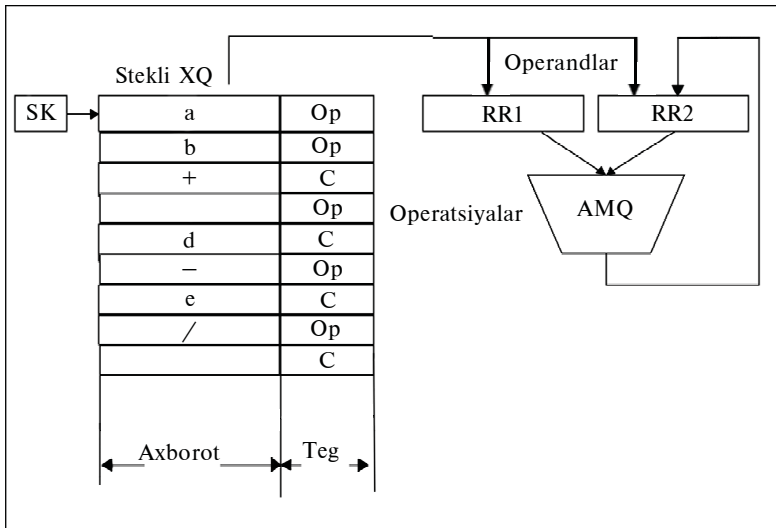
Agar buyruq adresi maydonning birinchi qismini bazali registr raqami belgilasa, uning ikkinchi qismi esa o‘zgartirish joylashgan registr raqamidan iborat bo‘lsa, bunday usul bazali-indeksli deyiladi

Stekli adresatsiya. Barcha zamonaviy mikroprotsessorlarni apparat darajasida stek qo‘llab-quvvatlaydi. Stek tezkor xotira-

ning sohasi bo‘lib, ixtiyoriy axborotni vaqtinchalik saqlash uchun mo‘ljallangan.

Xotirani stekli tashkil etishda buyruqlarni adressiz kodlashtirishni ko‘rib chiqamiz. Bunday xotiraning yacheykalariga murojaat etish ketma-ket maxsus stek ko‘rsatkichi (SK) orqali amalga oshiriladi. Stek ko‘rsatkichi joriy vaqtdagi ishchi yacheykani aniqlaydi. Har bir yacheyka teg bilan, ya’ni saqlanayotgan axborotning maxsus belgisi bilan jihozlangan.

Bunday EHM 6.18-rasmda namoyish etilgan strukturaga ega. Uning tarkibiga arifmetik-mantiqiy qurilmadan (AMQ) tashqari ikkita maxsus buferli registrlar (RR1 va RR2) kiradi. Bunda teglarning qiymatlari quyidagicha bo‘ladi: Op — ushbu yacheykada operand saqlanadi, C — yacheykada amal kodi mavjudligi belgisini bildiradi.



6.18-rasm. Adressiz buyruqlar tizimi bilan amallarni bajaruvchi EHM sxemasi.

Nazorat savollari va topshiriqlari

1. Xotirani tashkil etishning qanday usullarini bilasiz?
2. Bazali adresatsiya usuli qanday tashkil etiladi?
3. Kesh-xotiraning qanday afzallik tomonlari mavjud?
4. Nisbiy adresatsiyani tashkil etish texnologiyasini tushuntiring.

7-bob. ASSEMBLER TILI

7.1. ASSEMBLER TILI SAVIYASI

Biror-bir aniq dasturlash tilida yozilgan dasturlarni foydalanuvchi dasturlari ko‘rinishida tarjima qiladigan boshqa dasturlash tili *translatorlar* (tarjimonlar) deb ataladi. Dastur dastlab qaysi tilda yozilgan bo‘lsa, bu til *kiruvchi til*, dastur tarjima qilingan til esa *chiquvchi til* deyiladi. Kiruvchi va chiquvchi tillarni bosqichlar aniqlaydi. Agar protsessor kiruvchi tilda yozilgan dasturlarni ishlata olsa, bunda boshlang‘ich kodni qayta boshqa tilga tarjima qilish shart emas.

Tarjima qilish chiquvchi til uchun apparat yoki dasturli protsessor bor va kiruvchi tillar uchun esa protsessor bo‘lmaganda ishlatiladi. Agar tarjima to‘g‘ri qilingan bo‘lsa, tarjima qilingan dastur, dastlabki dastur natijasini beradi (agar unga to‘g‘ri keladigan protsessor mavjud bo‘lsa). Demak, dastlab chiqish bosqichida dasturlarni tarjima qiluvchi, so‘ngra olingan dasturlarni ishlatuvchi yangi bosqichni yaratish mumkin.

Tarjima qilish va interpretatsiya orasidagi farqlarni tushunish muhim hisoblanadi. Tarjima qilishda boshlang‘ich dastur kiruvchi tilda darhol ishlatilmaydi. Avval u obyektli dastur yoki ijrochi ikkilamchi dastur deb nomlanuvchi, tarjima to‘liq tugagandan so‘ng bajariladigan ekvivalent dasturga tarjima qilinadi.

Tarjima qilishda quyidagi ikki qadamdan o‘tish kerak:

1. Chiquvchi tilda ekvivalent dastur yaratish.
2. Olingan dasturni ishlatish.

Bu ikki qadam bir vaqtda ijro etilmaydi. Ikkinchi qadam faqatgina birinchi qadam tugashi bilan boshlanadi. Interpretatsiyada bitta qadam bor: bu boshlang‘ich dasturni bajarish. Hech qanday ekvivalent dasturni yaratish kerak emas, ammo ba’zida boshlang‘ich dastur interpretatsiyani soddalashtirish uchun (masalan, JAVA kodi) oraliq formaga aylanadi.

Obyektli dasturni bajarish vaqtida uchta bosqich qatnashadi: mikroarxitektura bosqichi, buyruqlar bosqichi va operatsion tizim bosqichi. Shunda dastur ishlash paytida kompyuter xotirasidan uchta dastur topish mumkin: foydalanuvchi obyektli dastur, operatsion tizim va mikroprogramma (agar u bo‘lsa). Shunda boshlang‘ich dasturning hech qanday izi qolmaydi. Shu tariqa, dasturni ishlatishda qatnashuvchi bosqichlar soni tarjima qilishdan oldingi bosqich sonidan farq qilishi mumkin.

Assembler tiliga kirish

Translatorlarni kiruvchi va chiquvchi tillar orasidagi munosabatlar bo‘yicha ikki guruhga bo‘lish mumkin.

Agar kiruvchi til sonli mashina tilining simvolli qayta tasvirlanishi bo‘lsa, ushbu translator assembler deb nomlanadi, kiruvchi til assembler tili deb nomlanadi. Agar kiruvchi til yuqori bosqichli til bo‘lsa (masalan, JAVA yoki C), chiquvchi til esa sonli mashina tili yoki oxirgi tilning simvolli qayta tasvirlanishi bo‘lsa, bu translator kompilator deb ataladi.

Assembler tili nima?

Assembler tili — bu har bir ma’lumot bitta mashina buyrug‘iga to‘g‘ri keladigan til. Boshqacha qilib aytganda, assembler tilida mashina buyruqlari va dasturdagi operatorlar orasida o‘zaro bog‘lanish mavjud. Agar assembler tilidan har bir qatorda bitta operator saqlansa va har bir mashina so‘zi bitta buyruqni saqlasa, assembler tilidagi dastur *n* qatorda mashina tilidagi *n* so‘zlardan dastur yaratadi.

Biz assembler tilini ishlatamiz, mashina tilida dasturlamaymiz (16 lik sanoq tizim hisobida), chunki assembler tilida dasturlash ancha oson. 2 lik va 8 lik sanoq sistemalari o‘rniga simvolik nom va manzillarni ishlatish ancha qulay.

Ko‘pchilik qo‘shish (add), ayirish (substract), ko‘paytirish (multiply), bo‘lish (divide) uchun ADD, SUB, ML va DIV buyruqlarni eslab qolish mumkin, lekin mashina ishlatadigan kerakli sonlarni eslab qolish qobiliyati yo‘q.

Dasturchiga faqat assembler tilidagi simvolik nomlarni bilish lozim, chunki assembler ularni mashina buyruqlariga aylantiradi. Bu manzillarga ham taalluqli. Dasturchi assembler tilida xotira yacheykalarning nomlarini berishi mumkin va darhol assembler unga to‘g‘ri sonlarni chiqaradi. Dasturchi mashina tilida har doim manzillarning sonli nomlari bilan ishlashi kerak. Hozir mashina tilida dastur yozadigan dasturchilar yo‘q, lekin bir necha o‘n yilliklar assembler tili ixtirolar qilingunga qadar, dasturlar shunday yozilgan.

Assembler tili yuqori bosqichli tillardan farq qiluvchi bir necha xususiyatlarga ega.

Birinchidan, assembler tili va mashina komandalari orasidagi bog‘lanishlar (bu haqda yuqorida aytib o‘tildi).

Ikkinchidan, dasturchi assembler tilida maqsadli mashinalardagi hamma obyekt va buyruqlarga ruxsat man etilmaydi. Yuqori

bosqichdagi tillarda ishlaydigan dasturchilarda bu imkoniyat yo‘q. Masalan, maqsadli mashina to‘lish bitiga ega, assembler tilidagi dastur uni tekshirishi mumkin. JAVA tilidagi dastur buni bajara olmaydi. Assembler tili dasturi maqsadli mashinaning buyruqlar to‘plamidagi har qanday buyruqni bajaradi, yuqori bosqich tili dasturlarida esa buning imkoni yo‘q.

Qisqacha qilib aytganda, mashina tilida qilib bo‘ladigan ishlarni assembler tilida qilish mumkin, lekin ko‘pchilik buyruqlar, registrlar va boshqa obyektlarni yuqori bosqichli tilda dasturchilar ishlata olmaydi.

Tizimli dasturlashdagi (masalan, C) tillar oraliq holatda bo‘ladi. Ular yuqori bosqich tili sintaksisiga ega, lekin ehtimollik nuqtayi nazaridan kirish man etilmaydigan assembler tiliga yaqin.

Nihoyat, assembler tilidagi dastur bir oilaga mansub kompyuterlarda ishlay oladi, yuqori bosqichli tilda yozilgan dasturlar potensial holda har bir mashinalarda ishlay oladi.

Dasturiy ta’minotni birinchi mashinadan ikkinchisiga o‘tkazish imkoniyati amaliy dasturlar uchun juda muhim hisoblanadi.

Assembler tili nima uchun kerak?

Assembler tili anchagina qiyin til. Assembler tilida dasturni yozish yuqori bosqichli tilda dasturni yozishga qaraganda ko‘proq vaqtni oladi. Bundan tashqari, sozlashga ko‘proq vaqtni egallaydi. Unda nega dasturlarni assembler tilida yozish kerak? Bunga ikkita sabab bor: samaradorlik va mashinaga to‘liq ruxsat borligi.

Birinchidan, tajribali dasturchi assembler tilida kichik hajmdagi dasturni yaratadi, bu dastur yuqori bosqichli tilda yozilgan dasturdan ancha tezroq ishlaydi. Ba’zi dasturlar uchun tezlik va hajm juda muhim mezon.

Ko‘pchilik birkitilgan amaliy dasturlar, masalan, kredit kartochkalar, uyali telefonlar, drayver uskunalari hamda BIOS protseduralari kabi dasturlar bu toifalarga kiradi.

Ikkinchidan, ba’zi protseduralarga apparat ta’minotiga to‘liq ruxsat kerak, buni yuqori bosqichli tilda yozilgan dasturda qilib bo‘lmaydi. Bu toifalarga operatsion tizimlardagi uzilishlar va uzilishlarga ishlov berish hamda tizimga kiritilgan real vaqtda ishlaydigan uskunalar nazoratchilari kiradi.

Birinchi sabab (yuqori ishlab chiqarish quvvatiga erishish) o‘ta muhim, shuning uchun uni chuqurroq ko‘rib chiqamiz.

Ko‘pchilik dasturlarda hamma kodlarning kichik foizi dastur bajarilgan vaqtning katta foizi uchun javob beradi. Ko‘pincha,

1% dasturlar 50% bajarish vaqti uchun, 10% dastur esa 90% bajarish vaqtiga tipik nazorat masalasini yechishda javob beradi. Masalan: yuqori bosqichli tilda dastur yozilishiga 10 odam/yil kerak bo‘lsa, ishlab chiqilgan dasturga 100 sekund kerak (nazorat masala — bu kompyuter, kompilator va boshq. taqqoslash uchun ishlatiladigan tekshirish dasturi). Assembler tilida yozilgan dastur yozilishiga 50 odam/yil ketishi mumkin.

Natijada olingan dastur nazorat ishini taxminan 33 sekundda yechadi, chunki yaxshi dasturchi kompilatordan uch marta aqlli bo‘lishi mumkin (izoh 7.1-jadvalda keltirilgan).

Faqat dasturning kichik qismi ushbu dasturni bajarish uchun ketgan vaqtning katta qismi uchun javob beradi, ammo boshqacha yo‘l tutish ham mumkin.

Avval dastur yuqori bosqich tilida yoziladi. So‘ngra dasturning qaysi qismi bajarilgan ishga ketgan vaqtning katta qismi uchun javob berishini aniqlash uchun qator o‘lchashlar olib boriladi. Bu o‘lchashlar uchun ko‘pincha tizimli taktli generator ishlatiladi. Uning yordamida har bir protsedura uchun qancha vaqt ketganligini, har bir sikl necha marta bajarilishini bilish mumkin.

Aytaylik, dasturning 10 % i uni bajarishga ketgan vaqtning 90 % iga javob beradi. Bu hamma 100 sekund ishning 90 sekundda 10 % i bajarilganligini, 10 sekund esa — qolgan 90 % dastur bo‘lishini anglatadi. Bu 10 foiz dasturni assembler tiliga qayta yozish bilan yaxshilash mumkin. Bu jarayon sozlash (tuning) deb nomlanadi. U 7.1-jadvalda keltirilgan. Asosiy protseduralarni to‘g‘rilash uchun yana besh yil kerak bo‘ladi, lekin dasturni bajarish vaqti 90 sekunddan 30 sekundgacha qisqaradi.

Assembler tili va yuqori bosqichli til ishtirok etgan, faqat assembler tili qo‘llaniladigan ushbu aralash yondashuvni taqqoslaymiz.

Ikkinchi usulda dastur 20 % tezroq ishlaydi (33 sekund 40 sekundga qarshi), lekin uch marta qimmat (50 odam/yil 15 ga qarshi).

Bundan tashqari, aralash yondashuvda yana bir ustunlik tomoni bor: yuqori bosqichli tilda yozilgan tayyor protsedurani, assembler tilida 0 dan yozgandan ko‘ra uni assembler kodiga aylantirish oson.

Agar dastur yozish bir yil vaqt talab etsa, assembler tili ishlatilgan aralash yondashuv va faqat assembler tilida yaratilgan

7.1-jadval

Assembler va yuqori bosqichli tillarda dasturlashni qiyoslash (sozlash va sozlashsiz)

	Dastur yaratishga ketgan odam/yil miqdori	Dasturni bajarishga ketgan vaqt, sekundda
Assembler tili	50	33
Yuqori bosqich tili	10	100
Sozlashgacha aralash usul: Kritik 10 % Qoldiq 90 %	 1 9 —	 90 10 —
Jami	10	100
Sozlashdan keyingi aralash usul: Kritik 10 % Qoldiq 90 %	 6 9 —	 30 10 —
Jami	15	40

dastur usuli orasidagi munosabat 4:1 nisbatda aralash yondashuv foydasiga hal bo‘ladi.

Yuqori bosqichli tilda ishlovchi dasturchilar bitlar ko‘chishi bilan shug‘ullanmaydi. Ular masalani yechish uchun dasturni shunday qurishi mumkinki, natijada samaradorlik haqiqatan yuqori pog‘onalarga ko‘tariladi. Assembler tilida yozadigan dasturchilar ko‘pincha bir necha sikllarni qisqartirish uchun buyruqlarni qurishga harakat qiladilar, shuning uchun ularda bu holat paydo bo‘lmaydi.

Biroq, bularga qaramay, assembler tilini o‘rganishning to‘rtta asosiy sabablari bor:

Birinchidan, assembler tilida dastur yozishni bilish maqsadga muvofiq hisoblanib, katta loyihaning omadli yoki omadsiz bo‘lishi ma’lum bir muhim protseduraning samaradorligini ikki yoki uch barobar oshirishga bog‘liq bo‘ladi.

Ikkinchidan, xotira yetishmovchiligi muammosidan chiqishning yagona yo‘li assembler tili bo‘lishi mumkin.

Kredit kartalar markaziy protsessorga ega, lekin ularda xotira megabayti va qattiq disk bo‘lmaydi. Ammo ular cheklangan resurslar borligida, qiyin hisoblashlarni bajarishi kerak. Elektrouskunalarga birlashtirilgan protsessorlar ko‘pincha minimal miqdordagi xotiraga ega bo‘lib, ancha arzon bo‘lishi lozim. Batareykalarda ishlaydigan har xil elektron uskunalar ko‘p hollarda kichik xotiraga ega, chunki bu yerda effektiv kod ham kerak bo‘ladi.

Uchinchidan, kompilator assembler ishlatadigan dasturning chiqish qiymatlarini berishi yoki assemblerlash jarayonini mustaqil bajarishi kerak.

Va nihoyat, assembler tilini o‘rganish real mashina to‘g‘risida tasavvur hosil qilish imkoniyatini beradi.

Nazorat savollari va topshiriqlari

1. Assembler tili nima?
2. Assembler tili nima uchun kerak?
3. Kompilator nima?

7.2. ASSEMBLER TILIDAGI OPERATOR FORMATI

Assembler tilidagi operator strukturasi berilgan mashina buyruq strukturasida aks etadi, har xil mashina va turli bosqichlardagi assembler tillari ko‘p tomonlama bir-biriga o‘xshash.

7.2—7.4-jadvallarda assembler tilida Pentium IV, Motorola 680×0 va (Ultra) SPARC uchun dasturlar lavhalari keltirilgan. Bu hamma dasturlar $N=I+J$ ifodani bajaradi. Uchala misolda dasturchilar jadvaldagi bo‘sh ifodalar ustida ish bajaradilar. Bo‘sh ifoda dasturi — o‘zgaruvchan *I*, *J* va *N* uchun assemblerda xotirani egallashidir. Bular mashina buyruqlarining simvolli qayta tasvirlanishi hisoblanmaydi.

7.2-jadval

N=I+J ifodani Pentium IV da qo‘llash

Belgi	Operatsiya kodi	Operand-lar	Kommentariylar
FORMULA	MOV ADD MOV	EAX,I EAX,J N,EAX	registr yeAX=I registr yeAX=I+J N=I+J

I	DW	3	4 baytni zaxiralaymiz va 3 ni o‘rnatamiz
J	DW	4	4 baytni zaxiralaymiz va 4 ni o‘rnatamiz
N	DW	0	4 baytni zaxiralaymiz va 0 ni o‘rnatamiz

7.3-jadval

N=I+J ifodani Motorola 680×0 da qo‘llash

Belgi	Operatsiya kodi	Operand-lar	Kommentariylar
FORMULA	MOVE.L ADD.L MOVE.L	I, D0 J, D0 D0,N	registr D0=1 registr D0=1+J N=I+J
1	DC.L	3	4 baytni zaxiralaymiz va 3 ni o‘rnatamiz
J	DC.L	4	4 baytni zaxiralaymiz va 4 ni o‘rnatamiz
N	DC.L	0	4 baytni zaxiralaymiz va 0 ni o‘rnatamiz

7.4-jadval

N=I+J ifodani SPARC da qo‘llash

Belgi	Operatsiya kodi	Operandlar	Kommentariylar
FORMULA	SETHI L.D SETHI L.D NOP	%HI(I),%R1 [%R1+%LO(I), %R1 %HI(J), %R2 [%R2+%LO(J), %R2	R1=I manzili katta bitlari R1=I R2=J manzili katta bitlari R2=J xotiradan J ni chaqiramiz
	ADD SETHI ST	%R1, %R2, %R2 %HI(N), %R1 %R2, [%R1+ +%LO(N)]	R2=R1+R2 R1=N manzili katta bitlari

1	WORD3	3	4 baytni zaxiralaymiz va 3 ni o‘rnatamiz
J	WORD4	4	4 baytni zaxiralaymiz va 4 ni o‘rnatamiz
N	WORD0	0	4 baytni zaxiralaymiz va 0 ni o‘rnatamiz

INTEL oilasi kompyuterlari uchun bir necha assemblerlar mavjud, ular bir-biridan sintaksis bo‘yicha farq qiladi. Bu kitobda Microsoft MASM assembler tilini ishlatamiz. Pentium IV kompyuterlari haqida so‘z yuritsak-da, ammo izohlarni hamma kompyuterlar uchun beramiz.

SPARC protsessorlari uchun assembler SUN ishlatiladi. Bularning hammasi oldinga 32 bitli versiyalarda ishlatiladi. Kitobda operatsiya kodi va ro‘yxatga olish hamma vaqt yozuv harflarida ifodalanadi, bu hol nafaqat Pentium IV assembler uchun, balki kichik harflarni ishlatuvchi SUN assembler tili uchun ham taalluqli.

Assembler tili to‘rt qatordan iborat: belgi qatori, operatsiya qatori, operand qatori va kommentariy qatori. Belgilar xotira manzillariga simvolik nomlarni ta’minlash uchun ishlatiladi. Ular buyruqlarga o‘tishni sodir qilish uchun kerak bo‘ladi. Bundan tashqari, ular simvolik qatorni saqlanadigan joyiga ruxsat olish mumkin bo‘lgan berilganlar so‘zlari uchun kerak bo‘ladi. Agar berilganlar belgilangan bo‘lsa, belgi ko‘pincha birinchi kolonkada joylashgan bo‘ladi.

Har bir uch masalada to‘rtta belgi bor: Formula, I, J va N. SPARC assembler tilida har qaysi belgidan keyin ikki nuqta qo‘yiladi, MOTOROLA uchun esa buning keragi yo‘q. INTEL kompyuterlari uchun ikki nuqta faqat buyruqlar belgilaridan keyin qo‘yiladi. Berilgan farqlar fundamental hisoblanmaydi. Turli assembler dasturchilari turli xil uslubga ega. Mashina arxitekturasi u yoki bu tanlovni aniqlamaydi. Ikki nuqtaning ustunligi, belgini alohida qatorda, operatsiya kodini esa — keyingi qator birinchi kolonkaga yozish mumkin. Bu kompilator ishini osonlashtiradi: ikki nuqtasiz boshqa qatordagi belgini operatsiya kodidan ajratib bo‘lmaydi.

Ba’zi assemblerlarda belgilar olti yoki sakkiz simvolgacha cheklangan. Ko‘pchilik yuqori bosqichli tilda nomlar uzunligi har xil bo‘lishi mumkin. Uzun va yaxshi tanlangan nomlar boshqa kishilar tomonidan dasturlarni o‘qish va tushunishni osonlashtiradi. Har bir mashinada bir nechta registrlar saqlanadi, lekin hammasiga umuman turli xil nomlar beriladi.

Pentium IV registrlari EAX, EVX, ESX, MOTOROLA registrlari D0, D1, D2 deb nomlanadi. SPARC mashinasidagi registrlar bir nechta nomga ega. Bu yerda ularni belgilash uchun % R1 va % R2 larni ishlatamiz.

Operatsiya kodi maydoni bu kodning simvolli abbreviaturasi (agar berilganlar mashina buyruqlari simvolik qayta tasvirlanishi hisoblansa) yoki assemblerning o‘zi uchun buyruqdan tashkil topadi. Nomni tanlash — dasturchining xohishi, shuning uchun assembler tilining har xil ishlab chiquvchilari ularni turli xil nomlaydilar. INTEL assembler ishlab chiqaruvchilari xotiradan registrni yuklash va xotirada registrni saqlash uchun MOV tushunchani ishlatadilar. MOTOROLA assemblerini ishlab chiqaruvchilar MOVE ni ikkala operatsiyada ishlatishni tanlashdi. SPARC assembler tilini ishlab chiquvchilari birinchi operatsiya uchun LD, ikkinchi operatsiya uchun ST ni ishlatadilar. Bunda mashina arxitekturasi nom tanlash bilan bog‘liq emas. Aksincha, SPARC arxitekturasi uskunasi ikki mashina buyrug‘ini xotiraga ruxsat uchun ishlatish kerakligini tushuntiradi, chunki virtual manzillari xotirasi o‘ttiz ikki bitli (xuddi SPARC Version8) va qirq to‘rt bitli (SPARC Version9) bo‘lishi, buyruqlari esa maksimum yigirma ikki bitni saqlashi mumkin. Demak, virtual manzilning hamma bitlarini uzatish uchun doim ikki buyruq talab qilinar ekan.

Buyruq:

SETHI %HI(I), %R1

Ushbu buyruq 64 razryadli R1 registrning katta o‘ttiz ikki bit va kichik o‘n bitlarini 0 ga aylantiradi, keyin 32 razryadli I o‘zgaruvchining katta yigirma ikki bitini R1 registrning o‘ndan o‘ttiz birgacha bitli pozitsiyalarga joylashtiriladi.

Keyingi buyruq:

LD[%R1+LO(I)],%R1

Ushbu buyruq R1 va I manzilning kichik 10 bitini qo‘shadi (natijada I to‘liq manzil hosil bo‘ladi), berilgan so‘zni xotiradan chaqiradi va uni R1 ga joylashtiradi.

Pentium, 680×0 va SPARC — oilalari protsessorlari hammasi har xil uzunlikdagi operandlarni (byte (bayt), word (so‘z), long (uzun) tipidagi) qabul qiladi. Assembler qanday qilib operandlar qanday uzunlikda ekanligini aniqlaydi? Turli assembler ishlab chiqaruvchilar turli xil yechimlarni qabul qildilar. Har xil

uzunlikdagi Pentium IV registrlari turlicha nomlanadi. 32 razryadli elementlarning siljishi uchun – EAX nomlari ishlatiladi, 16 razryadlilarda – AX, 8 razryadlilar uchun – AL va AH. MOTOROLA assembleri ishlab chiquvchilari har bir operatsiya kodiga qo‘shimcha qo‘shishni taklif etdilar: long tiplari uchun L, word tiplari uchun W va byte tiplari uchun B. Turli uzunlikdagi SPARC operandlari uchun har xil turdagi operatsiya kodlari ishlatiladi, masalan, baytni yuklash uchun yarimso‘z (halfword) va so‘z (word) 64 razryadli registrda mos ravishda LDSB, LDSH va LDSW operatsiya kodlari orqali ishlatiladi. Ko‘rib turganingizdek, tilni ishlab chiqarish ixtiyoriy olib boriladi.

Ko‘rib o‘tiladigan uchta assembler tillari, berilganlar uchun maydonni zaxiralash usuli bilan farqlanadi. INTEL uchun assembler tili ishlab chiquvchilar DW (Define Word — so‘zni topish)ni tanlashdi. Keyinchalik muqobil tarzda Word variant qabul qilindi. MOTOROLA da DC (Define Constant – konstantani topish) ishlatiladi. SPARC ishlab chiquvchilari boshdanoq Word ni tanladilar. Bundan ko‘rish mumkinki, so‘zlar farqi mutlaq ixtiyoriy olinadi.

Operandlar maydonida operatorga manzillar va registrlar beriladi. Butun sonni qo‘shish buyrug‘i operand maydonida nimani nimaga qo‘shishni bildiradi. O‘tish operandi maydonidagi buyruq, qayerga o‘tish kerakligini bildiradi. Operandlar bo‘lib — registrlar, konstantalar, xotira yacheykalari va hokazolar bo‘lishi mumkin.

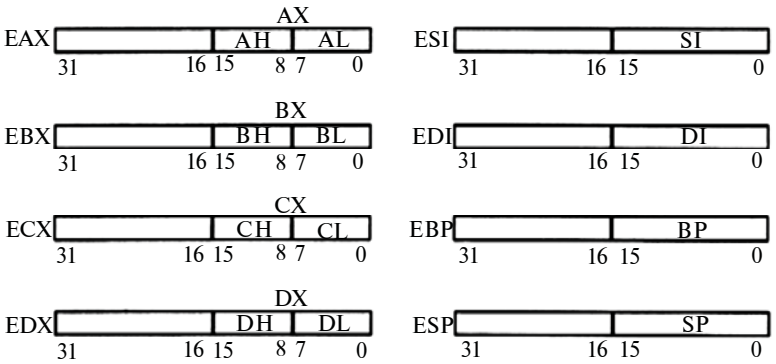
Kommentariy maydonida dasturlar holati haqida izohlar keltiriladi. Ushbu izohlar begona dasturni ishlatadigan va qo‘shimchalar qiladigan yoki dastur muallifining o‘zi bir yildan so‘ng uning ustida ishlaydigan dasturchilar uchun kerak bo‘lib qolishi mumkin. Assembler tilida dastur bu kommentariylarsiz dasturchilarga umuman tushunarsiz (shu dastur muallifiga ham). Kommentariylar insonlarga faqat foydali bo‘lishi mumkin. Ular dastur ishiga umuman ta’sir qilmaydi.

Nazorat savollari va topshiriqlari

1. Assembler qanday qilib operandlar qanday uzunlikda ekanligini aniqlaydi?
2. Assemblerda belgilar qanday ishlatiladi?
3. Assemblerda kommentariylarning o‘rni qanday?

7.3. PROTSESSOR VA XOTIRA

Protsessor registrlarining asosiylari quyidagilar: eax, ebx, ecx, edx, esi, edi, ebp, esp—32-razryadli registrlar. ax, bx, cx, dx, si, di, bp, sp — ularning kichik bo‘laklari (16 bitli so‘zlar). ax, bx, cx, dx-registrlari baytlar bo‘yicha kiritiladi, masalan, ah-ax katta bayti, al-ax kichik bayti. 32-razryadli registrlarning katta bo‘laklariga to‘g‘ridan to‘g‘ri kirib bo‘lmaydi (INTEL protsessorni soddalashtirish maqsadida shunday yo‘l tutgan bo‘ladi).



Har bir protsessor registri aniq bir maqsadga ega, registrlarni o‘z vazifasi bo‘yicha qo‘llash dasturni yaxshilashga olib keladi, ammo bu dalil emas. Registrlarning qayerda qo‘llanilishi quyida keltirilgan:

1) *eax* — akkumulator (harakat natijalarini saqlash);

2) *ebx* — baza registri (manzillashtirish uchun baza sifatida ishlatiladi);

3) *ecx* — hisoblagich sifatida ishlatiladi (sikllarda);

4) *edx* — berilganlar registri (eax dek);

5) *esi* — kelib chiqish indeksi;

6) *edi* — qabul qilish indeksi;

7) *ebp* — baza registri (kompilatordan ishlatiladi);

8) *esp* — stek registri (kompilatordan ishlatiladi).

Qaytarib aytish joizki, registrlarni xohlagancha va har qanday maqsadlarda *ebp* va *esp* lardan tashqari ishlatish mumkin (ular o‘ziga xos cheklovlar bilan ishlatiladi). Assemblerdagi qo‘yish *ebp* va *esp* registrlarni boshlang‘ich holatga keltirishi shart.

Assemblerda manzil masalasi va xotiraga yozish

1) *Manzil masalasi:* [S32+r32+1|2|4|8*(r32\esp)], bu yerda «S32» musbat, manfiy va nol bo‘lishi mumkin. Agar son 4 Gb dan ortiq bo‘lsa, bo‘lish ifodasidan qoldiq olinadi. Masalan, [eax] va [eax+4*esx], da yesx=1073741824 — ikkita bir xildagi manzil. Manzillashtirishning murakkab tipini ishlatsak, masalan, dword ptr [eax+4*ebx+10], bu qo‘rqinchli holat emas, bunday holatlarni protsessor yaxshi bajara oladi. U buni dword ptr [eax] ni ishlatishdagi tezlik bilan qariyb bir xilda bajaradi. Gap shundaki, protsessorda bu ko‘rinishdagi ifodalar ancha kuchli optimallashtirilgan. Bundan tashqari, bu turdagi protsessorlarning xususiyatlarini aniq bir tipdagi ifodalarni yechish uchun tez bajarishda ishlatish mumkin (ba’zida yordam beradi). Global o‘zgaruvchi manzilini offset yordamida olish mumkin. Masalan, «x» longint global o‘zgaruvchi tip bo‘lsa, unda «move ax, offset x», buyruqlaridan so‘ng, registr yeax «x» o‘zgaruvchi manzilni qabul qiladi. Agar keyinchalik «mov dword ptr [eax], 100» buyruq bajarilsa, «x» o‘zgaruvchi 100 qiymat ko‘rinishiga ega bo‘ladi.

2) *Xotira yacheykalari (o‘zgaruvchilar)ni yozish* paytida buyruqlarda quyidagilarni yozish mumkin:

byte ptr — 8 razryadli o‘zgaruvchi (Byte va shorint);

word ptr — 16 razryadli o‘zgaruvchi (integer va word tiplari);

dword ptr — 32 razryadli o‘zgaruvchi (int 64 va double);

tbyte ptr — 80 razryadli o‘zgaruvchi (extended tipi).

Agar *dword ptr x*, yozsak, «x» — 32 razryadli o‘zgaruvchi ko‘rinishga ega bo‘lamiz (*londint*, *single* va *x* tiplari).

3) *Intel qonuni:* Asosiy xotira qurilmasida 1 baytdan ko‘p bo‘lgan berilganlar quyidagicha saqlanadi: kichik bayt hamma vaqt kichik manzilda, shuning uchun massiv manzili — bu hamma vaqt faqat kichik yacheyka manzilida saqlanadi. Manfiy sonlar qo‘shimcha teskari kodda protsessorda saqlanadi, shuning uchun belgili va belgisiz qo‘shish va ayirish buyruqlari bir xil.

4) Intel protsessorlari dasturlarida stek yuqoridan pastga yo‘nalgan, berilganlarni kiritishda *esp* stek ko‘rsatkichi stekka kiritilgan bayt sonigacha kamayadi, stekda faqat ikki yoki to‘rt bayt kiritish mumkin (bir bayt mumkin emas).

Protsessor bayroqlari

ZF — nol bayrog‘i. Agar oldingi operatsiya natijasi nol bo‘lsa, 1 ga o‘rnatiladi.

SF — belgi bayrog‘i. U faqat natija bitining kattasiga teng.

SF — o‘tkazish bayrog‘i. Agar belgilangan boshlang‘ich operatsiya natijasi yoki (ayirishda) qarz kerak bo‘lganda, birga o‘rnatiladi, aks holda nolga teng bo‘ladi.

OF — to‘liq bajarish bayrog‘i. Belgili sonlar ustidagi arifmetik operatsiyalar boshlang‘ich chegarasidan o‘tmaydi va u birga o‘rnatiladi.

AF — yarimo‘tkazish bayrog‘i. Agar boshlang‘ich operatsiyalarda o‘tkazish yoki uch bitdan to‘rt bitga o‘tish qarzi bo‘lsa, birga o‘rnatiladi. Bu bayroq ikkilamchi-o‘nlamchi korreksiya buyruqlari bilan avtomatik ishlatiladi. Masalan, mov eax 15; inc eax buyruqlaridan keyin AF bayrog‘i birga teng bo‘ladi. Buyruqlar ketma-ketligidan so‘ng: mov eax 16; dec eax; AF bayrog‘i birga teng bo‘ladi. Buyruqlar ketma-ketligidan so‘ng: mov eax 16; dec eax; AF bayrog‘i ham birga teng bo‘ladi.

RF — juft bayroqlar. Agar boshlang‘ich buyruqlar mayda bayt natijalari birga teng bo‘lgan juft son bitlarini saqlasa birga o‘rnatiladi, aks holda nol bo‘ladi. Masalan, ikki buyruqda: mov al, 2; incal; RF — bayroq birga o‘rnatiladi.

IF — uzilishlar bayrog‘i. Bir uzilishlar ruxsat etiladi, 0 uzilishlar ruxsat etilmagan.

DF — yo‘nalish bayrog‘i. 0 qatorlar manzillar kattalashishi tomonga ishlanadi, bir manzillar kichrayishi tomoniga o‘zgarishi.

Nazorat savollari va topshiriqlari

1. Assemblerda manzil masalasi va xotiraga yozish qanday?
2. Intel qonuni nima?
3. Protsessor bayroqlari haqida nimalarni bilasiz?

7.4. ASSEMBLER ODDIY BUYRUQLARINING IFODALANISHI

1) *Mov qabul qilgich, manba* — qabul qilingan, berilganlarning o‘tish buyrug‘i hisoblanadi. Manbadagilarni qabul qilgichda saqlaydi, manbaning o‘zi o‘zgarmaydi. Masalan, mov ax, birinchi registrga ax ni yeax ga qo‘shadi. yeax manzildagi bayt kichik qismi ax (al)ga yoziladi, yeax+1 manzildagi bayt ah da yoziladi (Intel qonuniga asosan). Lekin buyruqlarni yozib, murakkab manzillashtirish shart emas, bizda o‘zgaruvchan «u» tipdagi longint bo‘lsa, keyingi buyruq yordamida 10000 ifodani olish mumkin: mov drowd ptr da 10000 ni yozish mumkin: «32» razryadli

o‘zgaruvchi ko‘rsatkichi ko‘rsatiladi. Agar mov drowd ptr [u+10], 5000 bo‘lsa, [u+10] manzildan boshlab xotiraga 32 bitli son 5000 yozib qo‘yiladi. Bularni yozishdan qo‘rqmaslik kerak (manzilga konstantani qo‘shish), chunki bu hamma ifodalar kompilatsiya bosqichida hisoblanadi va dastur tezligiga ta’sir qilmaydi. Masalan, keling «u» 34567 manzilda joylashgan bo‘lsin. Agar [u+10] ni yozib olsak, kompilator buni [32577]dek tushunadi, xuddi shunday «u» o‘zgaruvchining manzilini biladi. Mov buyrug‘ining operandlari ham registr, ham o‘zgaruvchi bo‘lishi mumkin, lekin bir vaqtning o‘zida ikki operand o‘zgaruvchi bo‘la olmaydi.

2) *Xshg operand1, operand2* — operandlarni almashtiradi. Masalan, agar al=45, ah=37, xchg al bajarilgandan so‘ng, ah al=37, ah=45 ga teng bo‘ladi.

3) *Add qabul qilgich, manba* — qabul qilgich va manbani birlashtirishni amalga oshiradi, natija qabul qilgichga kiritiladi. Manba o‘zgarmaydi, lekin bayroqlar o‘zgaradi.

4) *Adc qabul qilgich, manba* — qabul qilgich, manba va SF bayroq qo‘shishni bajaradi. Masalan, ikkita 64 bitli son bor, birinchisi edx:eax da (kichik ikkilamchi so‘z — yeax da), katta ikkilamchi so‘z edx — da). Ikkinchisi yebx:esx da. Shunda, add eax, yesx; adc edx, ebx buyruqlarni bajarishdan keyin, edx:eax registrlari ushbu 64 bitli sonlar yig‘indisini saqlaydi. 16 razryadli protsessorlar bo‘lganda, 32 bitli sonlar qo‘shiladi (ularning har biri ikkita 16 bitli sonlardan iboratligini hisoblagan holda). Hozir bu ahamiyatga ega emas. Bundan tashqari, protsessorga 16 razryadli yoki 32 razryadli sonlarni qaysisini qo‘shishning farqi yo‘q. Tezlik bir xil saqlanadi. Shuning uchun 32 bitli sonlarning to‘g‘ridan to‘g‘ri qo‘shish ularni 16 bitliga ajratib qo‘shishdan ko‘ra (2 marta) tez bo‘ladi. Ammo, ba’zida bu kerak bo‘lib qoladi. Masalan, Turbo-Paskal 32 bitli registrlarni tushunmaydi. Agar Turbo-Paskalda longint o‘zgaruvchisini kiritsak, operatsiyalar yuqori saviyada bajarilmaydi. Bu bilan 16 bitli dasturlar 32 bitlidan sekinroq ishlaydi.

5) *Sub qabul qilgich, manba* — qabul qilgichdan manbani ayiradi, natijani qabul qilgichga o‘zlashtiradi.

6) *Sbb qabul qilgich, manba* — qabul qilgich manbadan berilganlarni ayiradi, keyinchalik SF ni ayiradi. Uni 64 bitli so‘zlarni ayirish uchun ishlatish mumkin.

7) *Ins qabul qilgich* — add qabul qilgich 1 ga o‘xshaydi, ya’ni qabul qilgichni bittaga oshiradi.

8) *Des qabul qilgich* — sub qabul qilgich 1 ga o‘xshaydi, ya’ni qabul qilgichni bittaga kamaytiradi.

9) *Cmp operand1, operand2* — operand2 ni operand1 dan ayirishda, operandlar o‘zgarmaydi, buyruq faqat bayroqlarni o‘zgartiradi. Bu buyruq yordamida, odatda, shartli o‘tishlar amalga oshiriladi (eng tarqalgan usul).

10) *And/or qabul qilgich, manba* — qabul qilgich, manba VA/YOKI mantiqiy bit bo‘yicha bajaradi va natijani qabul qilgichga kiritadi. Ko‘pincha alohida bitlarni birga aylantirib, tanlab olish yo‘li bilan 0 ga aylantirish maqsadida ishlatiladi. Masalan, and al, 00001111b al registrning katta 4 bitini nolga aylantiradi, kichiklari esa o‘zgartirmaydi.

11) *Xor qabul qilgich, manba* — mantiqiy YOKI ning istisno bo‘lishi, qabul qilish va manba ustida mantiqiy YOKI ning bitlari istisno bo‘lishi, natija qabul qilgichga o‘zlashtiriladi. Ko‘pincha registrlarni nolga aylantirish uchun ishlatiladi. Masalan, xor ax, ax registrni nolga aylantiradi va buni mov ax, 0 ga qaraganda tezroq bajaradi. Bu buyruq yordamida registrlarni 0 ga aylantirish amalga oshiriladi. Buyruq Intel kompyuterlarida yaxshi ishlaydi. Buyruq har qanday Intel tomonidan registrni nolga aylantirish sifatida rasmiy qo‘llaniladi.

12) *Test operand1, operand2* — and operandlari ustida bajariladi, lekin operandlar o‘zgarmaydi, faqat and bayroqlari o‘zgaradi.

13) *Not qabul qilgich*, har bir qabul qilgichning nolga teng biti birga va aksincha, birga teng biti 0 ga almashtiriladi. Bayroqlar o‘zgarmaydi.

Bayroqning asosiy buyruqlari

Quyida keltirilgan jadval, protsessorning asosiy buyruqlarini qanday yozish mumkinligini tushunishda yordam beradi (masalan, agar buyruq ikki operandli bo‘lsa, unda ikki operand xotira yacheykalari bo‘la olmasligi unda ko‘rsatib o‘tilgan). Agar buyruq unarli (parametrlar soni 1 ga teng) bo‘lsa, unda yagona operand nimaligini ko‘rsatib o‘tishimiz kerak (registr yoki o‘zgaruvchi). Agar «r» qatorda «+» bo‘lsa, demak, mumkin va aksincha, «—» bo‘lsa, demak, mumkin emasligini bildiradi. Agar bo‘sh bo‘lsa, u holda ahamiyatga ega emas (ikki operand buyruqlarining yagona operandi nimani tashkil qilishi haqida gapirish ahamiyatsiz). Agar buyruq binar bo‘lsa, jadval operandlarning qandayligini ko‘rsatadi.

Hamma jadvaldagi binar buyruqlar operandlari bir xil razryadga ega bo‘lishi kerak. (8 | 16 | 32 | 64). ZF, SF, CF, OF, AF, PF — bayroqlari operatsiyasi o‘zgaradimi? «+» — o‘zgaradi, bo‘sh — o‘zgarmaydi, «?» — aniqlay olmaydigan qiladi; «=0» — nolga aylantiradi.

Buyruq	Para-metr	r	m	rr	rm	mr	rC	mC	ZF	SF	CF	OF	AF	PF
mov x,y	2			+	+	+	+	+	–	–	–	–	–	–
xchg x,y	2			+	+	+	–	–	–	–	–	–	–	–
add x,y	2			+	+	+	+	+	+	+	+	+	+	+
ads x,y	2			+	+	+	+	+	+	+	+	+	+	+
sub x,y	2			+	+	+	+	+	+	+	+	+	+	+
sbb x,y	2			+							+	+	+	+
inc x,y	1	+	+						+	+	–	+	+	+
dec x,y	1	+	+						+	+	–	+	+	+
cmp x,y	2			+	+	+	+	+	+	+	+	+	+	+
and|or|xor|test x,y	2			+	+	+	+	+	+	+	=0	=0	?	+
not x	1	+	+						–	–	–	–	–	–

1) *Push* r16 | r32 | m16 | m32 | C16 | C32 ni stekka joylashtirish. Bayroq o‘zgarmaydi. «push sp|esp» — esp ifodasi kamaygunigacha stekka joylashtiradi. Konstantani stekka kiritish uchun, undan oldin berilgan manzildan necha bayt stekka kiritish kerakligini kompilator tushunishi uchun «word ptr» yoki «drowd ptr»ni yozish kerak. Stekka so‘zni (2 baytli), ikkitali so‘zni (4 baytli) kiritish mumkin, lekin bir baytli emas.

Ror r16| r32| m16| m32 — stekdan chiqqan berilganlarni o‘qish. Bayroqlar o‘zgarmaydi. Agar operand-xotira yacheykasi, manzillashda esp dan foydalansa, unda ror buyrug‘i operand esp oshirishdan keyin manzilini topib oladi.

2) *Lea* r32, [manzil] – 32 razryadli registr manzillar berilganlarni yozib oladi. Bayroqlar o‘zgarmaydi. Manzillar kvadrat qo‘shtirnoqlarda yoziladi. Shunday qilib, ba’zi bir o‘zgaruvchilarning qiyin bo‘lgan manzillarini aniqlash mumkin. Masalan, bizda y: array [0..1000] of longint global massiv berilgan bo‘lsin. Unda lea eax, [e+4*200] buyruqni bajarishdan keyin, yeax registrida y[200] yacheykaning manzili saqlanadi (noldan boshlab

200 — element massivigacha). 4 baytli berilganlar massivi bo‘lganligi uchun 4 ga ko‘paytirildi. *Lea* buyrug‘ini aniq bir tip ko‘rinishlarini tez hisoblash uchun ishlatish mumkin.

3) *Cbw* — al ni ax gacha kengaytiradi;
Cwd — ax ni dx : ax gacha kengaytiradi;
Cwde — ax ni eax gacha kengaytiradi;
Cdq — eax ni edx eax gacha kengaytiradi.

Birliklarni kengaytirish agar boshlang‘ich registrning katta biti bir bo‘lsa amalga oshiriladi, aks holda nollar kengayadi. Aslida, ishorali sonlar kengaytiriladi. Masalan, bizda shortint tipidagi 00111101b (demak, 61) son berilgan bo‘lsin. Buning ustida cbw buyrug‘ini bajarganimizdan keyin, u bizga 0000000000111101 (ya’ni 61) sonini qaytaradi. Ammo bizda manfiy son bo‘lganda, masalan, 10001000b (demak − 120) uni kengaytirgandan keyin 1111111110001000 b (ya’ni 120) soni hosil bo‘ladi. Shunday qilib, bu buyruqlar ishorali sonni kengaytiradi. Ishorasiz sonlarni kengaytirish uchun qo‘shimcha qismini nollar bilan to‘ldirish kerak bo‘ladi, buni bajaradigan buyruqlar mavjud.

4) *movsx* qabul qilgich, manba − manbaning tashkil etuvchilarini (r8_yoki_m8 | r16_yoki_m16) qabul qilgichga (r16|r32) ko‘chiradi va cbw|cwde buyruqlari singari ishorani kengaytiradi. movzx buyrug‘i manbaning tashkil etuvchilarini (r8_yoki_m8 | r16_yoki_m16) qabul qilgichga (r16|r32) ko‘chiradi va nollar bilan kengaytiradi. Aslida ishorasiz sonlarni kengaytiradi.

5) *Bswap* r32 − 32 razryadli registrda birinchi bilan to‘rtinchi baytning va ikkinchi bilan uchinchi baytning joylari o‘zaro almashtiriladi. Masalan, agar registr yeax=12345678h bo‘lsa, bswap yeax buyrug‘i bajarilgandan so‘ng yeax=78563412 ga teng bo‘ladi.

6) *xlatb* − jadval bo‘yicha tarjima qilinadi. Xotira jadvalidan al baytni ebx manzili bo‘yicha jadval boshidagi farqli o‘tkazish bilan kiritiladi.

7) Masalan,

```
jmp @begin;
@htable: db '0123456789ABCDEF';
@begin:
  mov al, 09h;
  mov ebx, offset @htable;
  xlatb;
```

Bundan keyin al=Ord (9) (ASCII-9 soniga, demak, al=39 h) ga teng bo‘ladi.

8) *In qabul qilgich, manba* – qabul qilgichga (al | ax | eax) kiruvchi-chiquvchi portidan manbada ko‘rsatilgan raqamni ko‘chiradi. Manba – dx registr yoki 8 bitli konstanta bo‘lishi mumkin.

9) *Out qabul qilgich, manba* – (registr dx yoki 8 bitli konstanta) manbada ko‘rsatilgan port sonidan sonni al | ax | eax qabul qilgichga nusxalaydi. In va out ko‘pincha (rul, joystik, printer, skaner, ...) turli uskunalar drayverlari tomonidan ishlatiladi. Yana ular uzilishlarni qayta ishlovchilar tomonidan ham ishlatiladi. Int 21h ni yozganimizda, haqiqatan ham juda ko‘p jarayonlar (shu qatorda in va out bajarilishlar to‘plami) bo‘lib o‘tadi.

10) *Imul manba* – ishorali ko‘paytirish. Manba r8 | r16 | r32 | m8 m16 | m32. Manba operand razryadiga bog‘liq holda al | ax | eax ga ko‘paytiriladi va natija ax| dx : ax| yedx : yeax ga mos ravishda joylashtiriladi. Agar natija kichik bo‘lakka sig‘sa, unda CF=OF=0, aks holda, SF=OF=1 bo‘lib saqlanadi. ZF, SF, AF, PF bayroqlari aniqlanmagan. Demak, mul/imul ko‘paytirish buyrug‘i, natijaning 2 marta uzun to‘plamlar bo‘lishini ma’lum qiladi. Bu ancha murakkab buyruq.

11) *Mul manba* – ishorasiz ko‘paytirish.

12) *Idiv manba* – ishora bilan butun sonli bo‘lish. Manba r8 | r16 | r32 | m8 | m16 | m32. Ishorali butun sonli bo‘lishni quyidagicha amalga oshiradi:

8 bit: ax / x8; al – xususiy; ah qoldiq.

16 bit: dx : ax / x 16; ax – xususiy; dx qoldiq.

32 bit: yedx : yeax / x 32; yeax – xususiy; yedx qoldiq.

ZF, SF, CF, OF, AF, PF bayroqlari bu buyruqdan so‘ng aniqlanmagan. Xatarli tarafi: to‘lib-toshish va nolga bo‘lishda programma yo‘qoladi! Masalan, agar ax=256, cl=1, unda div cl buyruqlari al da 256 soni bilan yozilishi kerak, bu esa bo‘lishi mumkin emas (8 bitli registr bo‘lganligi sababli). Shuning uchun dastur yo‘qoladi. Xuddi shunday dividiv buyruq ham ancha murakkab hisoblanadi.

13) *Div manba* – ishorasiz bo‘lish (oldingisiga o‘xshash).

14) *Imul qabul qilgich, manba* – manba (c | r | m) qabul qilgichga (registr) ko‘paytiriladi, natija qabul qilgich (registr)ga kiritiladi. Operand razryadi 16 yoki 32.

Imul qabul qilgich, manba1, manba2 – manba1 (r | m) manba2 (son)ga ko‘paytiriladi, natija qabul qilgich (registr)ga kiritiladi.

Agar to‘lib-toshish sodir bo‘lsa, natijada katta bit yo‘qotiladi, unda OF = CF =1, aksincha, OF = CF = 0 bo‘ladi. SF, ZF, AF, PF ifodalari aniqlanmagan.

15) *Neg qabul qilgich* – qabul qilgichda saqlanadigan sonlar ustida (r8 | r16 | r32 | m8 | m16 | m32) ikkigacha to‘ldirish operatsiyalarni bajaradi. Bu operatsiyalar agar qabul qilgichga xuddi ishorali son sifatida qaralsa, ishoraga murojaatga teng kuchli bo‘ladi. Agar qabul qilgich nolga teng bo‘lsa, unda CF=0, aksincha, CF=1 bo‘ladi.

O‘z navbatida, qolgan bayroqlar (OF, SF, ZF, AF, PF) operatsiyalar natijalari qilib tayinlanadi. Agar bayroq unutilsa, haqiqatda neg buyrug‘i not va inc (qo‘shimcha teskari kodda manfiy sonlar qanday saqlanishini yodga olamiz) buyruqlarining bajarilish ketma-ketligiga to‘g‘ri keladi.

neg buyrug‘ini ishlatish bo‘yicha eng yaxshi misol – son ifodasining absolut qiymatini olish: @label; neg eax; js@label.

14) *Pushad* – umumiy holdagi registrlar stekiga joylashtiradi. Registr stekiga joylashtirish quyidagi ketma-ketlikda amalga oshiriladi: eax, ecx, edx, ebx, esp, ebp, esi, edi (edi stek yuqorisida bo‘ladi). Esp ifodasi buyruq ish bajarishdan oldin ishlatiladi.

15) *Popad* – umumiy holdagi registr stekni yuklash. Qarama-qarshi pushad harakatini bajaradi, lekin stekda joylashgan esp yo‘qoladi.

16) *Xadd qabul qilgich, manba* – o‘zaro almashinish va qo‘-shish. Qo‘shishni bajaradi, qabul qilgichni manbaga joylashtiradi, jami — qabul qilgichga joylashtiriladi. Manba registr, qabul qilgich registr yoki o‘zgaruvchi. Operandlar razryadligi –8 | 16 | 32.

xadd	Oldingi	Keyingi
Qabul qilgich	X	X+U
Manba	U	X

17) *Cmpxchg qabul qilgich, manba* – o‘zaro taqqoslash va qo‘shish. Qabul qilgich (registr) bilan al | ax | eax ifodalarini taqqoslaydi. Agar u qabul qilgichga teng bo‘lsa, qabul qilgich: = manba va ZF: = 1, aks holda al | ax | eax: = qabul qilgich va ZF: = 0 bo‘ladi.

Qolgan bayroqlar taqqoslash operatsiyalari natijalari bo‘yicha o‘rnatiladi, smr dan keyin bo‘lganidek. Manba — registr, qabul qilgich – r/m.

18) *smrxchg8b qabul qilgich* – 8 baytli taqqoslash va almashtirish. yedx:eax qabul qilgich bilan (m64) taqqoslanadi. Agar ular teng bo‘lsa, unda qabul qilgich: =esx:ebx. Aks holda yedx : yeax : = qabul qilgich.

Nazorat savollari va topshiriqlari

1. Bayroqning asosiy buyruqlari.
2. MOV operatorining vazifasi nima?
3. PUSH va POP buyruqlari nima uchun ishlatiladi?

7.5. DIREKTIVALAR

Assembler tilidagi dasturlar nafaqat protsessor bajaradigan mashina buyruqlarini, balki assembler o‘zi bajaradigan buyruqlarni ham aniqlashi kerak (masalan, ozgina xotirani ajratishi yoki listingning yangi sahifasini chiqarishi kerak).

Assembler buyruqlari soxta buyruqlar yoki assembler direktivalari deb nomlanadi. Biz 7.2-jadvalda o‘ziga xos bo‘lgan DW soxta buyruqni uchratgan edik. 7.5-jadvalda ba’zi bir boshqa soxta buyruqlar (direktivalar) keltirilgan. Ular Intel oilasi uchun MASM assembleridan olingan.

7.5-jadval

MASM assembleri ba’zi bir direktivalari

Direktivalar	Xususiyati
SEGMENT	Yangi segmentni (matn, berilganlar va boshqalar) aniq bir atributlari bilan boshlaydi
ENDS	Joriy segmentni tugatadi
ALIGN	Keyingi buyruq va berilganlarni to‘g‘rilab nazorat qiladi
EQU	Berilgan ifodaga to‘g‘ri keladigan yangi simvolni aniqlaydi
DB	Bir yoki bir necha bayt uchun xotira bajaradi
DD	16 bitli yarim so‘z uchun bir yoki bir necha bayt xotira bajaradi
DW	32 bitli yarim so‘z uchun bir yoki bir necha bayt xotira bajaradi
DQ	64 bitli yarim so‘z uchun bir yoki bir necha bayt xotira bajaradi

Keyingi 4 direktivalar DB, DD, DW, DQ 1, 2, 4 va 8 bayt o‘lchamli bir yoki bir necha o‘zgaruvchilarga mos ravishda xotira ajratadi. Masalan,

TABLE DB 11, 23, 49

Ushbu direktiva 3 bayt uchun joy ajratadi va ularga boshlang‘ich qiymat sifatida 11, 23 va 49 ko‘rinishlarni birlashtiradi. Bundan tashqari, bu direktiva TABLE simvolni aniqlaydi va shu manzilga teng 11 sonini saqlaydi.

PROC va ENDP direktivalari assembler tili protseduralari boshlanishi va yakunlanishini aniqlaydi. Assembler tilidagi protseduralar ham yuqori darajali dasturlash tillaridagidek vazifani bajaradi. MACRO va ENDM direktivalari makrosning boshlanishi va yakunlanishini aniqlaydi. Makroslar haqida keyinroq to‘xtalib o‘tamiz.

Keyinchalik PUBLIC va EXTERN direktivalari keladi. Ko‘pincha dasturlar fayllar to‘plami ko‘rinishida yoziladi. Ba’zida bir faylda joylashgan protsedurani boshqa faylda chaqirib ishlatish yoki berilganlarga ruxsat olishga to‘g‘ri keladi. Bunday fayllar orasidagi havolalar o‘rnatish imkoniyati vujudga kelishi uchun, ko‘rsatkich (nom), boshqa fayllarga qo‘llash uchun PUBLIC direktivalari yordamida eksportlanadi. Assembler ushbu faylda noma’lum bo‘lgan simvolning ishlatilishi bo‘yicha ogohlantirish chiqarmasligi uchun ushbu simvol tashqi (EXTERN), ya’ni aniq bir boshqa faylda e’lon qilingan bo‘lishi mumkin. Biror-bir direktivalarda aniqlanmagan simvollar, faqat bir fayl chegarasida ishlatiladi. Shuning uchun, agar simvol F00 bir nechta fayllarda ishlatilsa, bu hech qanday mojaroga olib kelmaydi, bu simvol har bir faylga nisbatan lokal hisoblanadi.

INCLUDE direktivasi assemblerga boshqa faylni chaqirishga buyruq beradi va faylga qo‘shadi. Bu kiritilgan fayllar, odatda, har xil fayllar uchun kerakli aniqliklarga ega makros va boshqa elementlarni o‘zida saqlaydi.

Assemblerning ko‘pgina tillari, shu qatorda MASM dasturning shartli komponovkasini qo‘llab turadi.

Masalan, dastur

```
WORDSIZE EQU 16
IF WORDSIZE GT 16
WSIZE: DW32
ELSE
WSIZE: DW 16
ENDIF
```

Ushbu dastur xotiradan bitta 32 bitli so‘z uchun joy ajratadi va uni WSIZE manzili orqali chaqiradi. Bu so‘zga qiymatlardan biri beriladi: WORDSIZE qiymatiga bog‘liq yo 32 yoki 16 ko‘rsatkichlari kiritiladi (bu holda 16). Bu qurilma 16 razryadli mashinalar uchun dasturda ishlatilishi mumkin (8088 dek) yoki 32 razryadli mashinalar uchun (Pentium IV dek). Agar mashinaga tobe kodning boshida va oxirida IF va ENDIF kodini qo‘shsak, undan keyin WORDSIZE qiymatini yagona aniqlikni o‘zgartirish, dasturda ikki o‘lchamdan birini avtomatik sozlash imkoniyati mavjud bo‘ladi. Bu usul orqali, bir nechta har xil mashinalar uchun bitta dasturni ishlatish mumkin. Ko‘pchilik hollarda hamma mashinaga tobe aniqlashlar, xuddi WORDSIZE dek, bir faylda saqlanadi va turli xil mashinalar uchun har xil fayllar bo‘lishi kerak. Fayllarni kerakli aniqliklar bilan kiritish yo‘li orqali dasturni osonlik bilan har xil mashinalarga qayta qurish (kompilatsiya qilish) mumkin.

COMMENT direktivasi foydalanuvchiga kommentariy boshidagi simvol (nuqta-vergul)ni a’lo darajada o‘zgartirishga yo‘l qo‘yadi. PAGE direktivasi dastur listingini boshqarish uchun ishlatiladi. Nihoyat, END direktivasi dasturning tugaganligini bildiradi.

MASM assemblerida direktivalar ko‘p. Pentium IV uchun boshqa assemblerlar boshqa qator direktivalarni saqlaydi, ular mashina arxitekturasi bilan aniqlanmay, assembler ishlab chiqaruvchilar xohishi bilan aniqlanadi.

Nazorat savollari va topshiriqlari

1. Soxta buyruqlar nima?
2. Assemblerda tashqi fayllarni qo‘shish qanday amalga oshiriladi?

7.6. MAKROSLAR

Assembler tilida ishlaydigan dasturchilar ko‘pincha bitta yoki bir necha buyruqlar ketma-ketligini qayta-qayta amalga oshirishlariga to‘g‘ri keladi. Kerakli buyruqlarni qachon ular talab qilinsa yozish eng osonroq tuyuladi. Lekin ketma-ketlik ancha uzun yoki ko‘p marta qaytarish kerak bo‘lsa, unda bu ancha murakkab bo‘ladi.

Muqobil usul – ushbu ketma-ketlikni protsedura ko‘rinishida yozish va kerakli paytda unga murojaat qilish kerak. Bu strategiyaning o‘z kamchiliklari bor, ushbu holatda har safar

maxsus protseduralarni — chaqiruv buyrug‘i va qaytarish buyrug‘ini bajarish kerak bo‘ladi.

Agar buyruqlar ketma-ketliklari hamma vaqt qisqa bo‘lsa-yu, (masalan, faqat ikki buyruq), lekin tez-tez ishlatilsa, protsedura chaqiruvchi dasturning ishlash tezligini juda pasaytiradi. Ushbu muammoning sodda va samarali yechimi makroslar bo‘ladi.

Makroaniqlashtirish, makrochaqiruv va makrokengayish

Makroaniqlashtirish − matnning bir qismiga nom berish usulidir. Bundan keyin makros aniqlanadi, dasturchilar dasturning bir qismi o‘rniga makrosning nomini yozadi. Makros − matnning bir qismi bo‘lib hisoblanadi.

7.1-listingda assembler tilidagi dastur Pentium IV uchun keltirilgan, ular *r* va *q* o‘zgaruvchilarning joylarini ikki marta almashtiradi. Bu buyruqlar ketma-ketligi makroslardek aniqlanadi (7.2-listing). Makroslarni aniqlashdan keyin har bir dasturdagi SWAP nomi quyidagi to‘rt qator bilan to‘ldiriladi:

```
MOV yeAX.P
MOV yeVX.Q
MOV Q.EAX
MOV P.EBX
```

Dasturchi SWAP ni ushbu 4 qator operatorlar uchun aniqlaydi.

Assemblerning har xil tillari makroslarni aniqlash uchun har xil yozuvlar ishlatadi, ularning barchasi bir xil baza qismlaridan iborat:

1. Makros so‘zboshi, aniqlanadigan makros nomi berilgan.
2. Makros keltirilgan matn.
3. Aniqlashni yakunlovchi direktiva (masalan, ENDM).

Assembler dasturdagi makroaniqlashga to‘qnashsa, keyinchalik ishlatish uchun uni jadvalga saqlaydi. Dasturda operatsiya kodi sifatida makros paydo bo‘ladi (bizning holda SWAP), assembler uni makros tanasi bilan almashtiradi. Makros nomini ishlatish operatsiya kodi sifatida makro chaqiruv deyiladi, uning makros tanasi bilan almashishi makro kengayish deb nomlanadi.

7.1-listing

Assembler tilidagi kod, *p* va *q* o‘zgarishlari ikki marta joy almashinadi (makrosning ishtirokisiz)

```
MOV yeAX.P
MOV yeVX.Q
```

```
MOV Q.EAX
MOV P.EBX
MOV yeAX.P
MOV yeVX.Q
MOV Q.EAX
MOV P.EBX
```

7.2-listing. Xuddi shu kodli makros ishlatiladi:

```
SWAP    MACRO
        MOV yeAX.P
        MOV yeVX.Q
        MOV Q.EAX
        MOV P.EBX
        ENDM
        SWAP
        SWAP
```

Makrokengayish assemblerlashtirish jarayoni vaqtida sodir bo‘ladi, dastur bajarilishi paytida emas. Bu juda muhim. 7.1 va 7.2-listinglardagi keltirilgan dasturlar shu bitta mashina kodini hosil qiladi. Mashina tilida dastur bo‘yicha boshlanishida makroslar ishlatilganligini aniqlab bo‘lmaydi. Makrokengayish tugagandan so‘ng, assembler makrokengayishni cheklab o‘tadi. Hosil qilingan dasturlarda hech qanday makros xususiyati qolmaydi.

Makrochaqiruvlarni protsedurani chaqiruvchi bilan tenglashtirish yaramaydi. Asosiy farq, makrochaqiruv bu assembler buyrug‘idagi makros nomini o‘zi bilan almashtirishdir. Protsedura chaqiruvi – mashina buyrug‘i, obyekt dasturga kiritilgan va protsedurani chaqirish uchun keyinchalik bajariladi.

7.6-jadvalda makrochaqiruv va protsedura chaqiruvi taqqoslanadi.

7.6-jadval

	Makrochaqiruv	Chaqiruvchi protsedura
Dastur chaqiruvchi qachon bajariladi?	Assemblerlashtirish vaqtida	Bajarish vaqtida
Obyektli dasturda har safar makros tanasi yoki protsedura o‘yiladimi?	HA	YO‘Q

Protsedura chaqiruvi buyrug‘i obyektli dasturda oldin kiritilganidan keyin bajariladimi?	YO‘Q	HA
Chaqiruvdan keyin qaytarish buyrug‘idan foydalanish mumkinmi?	YO‘Q	HA
Obyektli dasturda qancha makrochaqiruv yoki protsedura tanasining nechta nusxasi paydo bo‘ladi?	Makrochaqiruv 1 ta	1

Assemblerlash jarayoni ikki yo‘l bilan amalga oshiriladi. Birinchi yo‘lda makroaniqlashlar saqlanadi, makrochaqiruvlar esa kengaytiriladi. Ikkinchi yo‘lda natijada olingan matnga ishlov beriladi. Boshqacha qilib aytganda, oxirgi dastur chiqariladi, so‘ng boshqa dasturga aylanadi, undan hamma makroaniqlashlar yo‘qotilgan, har biri chaqiruv makros bilan almashadi. Olingan dasturlar makroslarsiz keyinchalik assemblerlarga o‘tadi. Muhimi, shuni inobatga olish kerakki, dastur qator simvollardan iborat. Bular harf, son, probel, punktuatsiya belgilari va qisqa qaytishi (yangi qatorga o‘tish). Makrokengaytmaning ishi joriy zanjirdagi (zanjir bu belgilar to‘plami) ma’lum bir zanjirlarni boshqa zanjirlar bilan almashtirishdan iborat. Makrovositalar – bu belgilar zanjirini ularning qiymatiga bog‘liq bo‘lmagan holda boshqarish usulidir.

Nazorat savollari va topshiriqlari

1. Makroslar nima?
2. Makroslarning protseduralardan farqi nima?
3. Makrokengaytma nima?

FOYDALANILGAN ADABIYOTLAR

1. G‘aniyev S. K. Elektron hisoblash mashinalari va sistemalari. – T.: «O‘qituvchi», 1990.

2. Максимов Н. А. Архитектура ЭВМ и вычислительные системы. — М.: «ФОРУМ-ИНФРА-М», 2005.

3. Соловьев В. А. Схематика ЭВМ. – М.: «Высшая школа», 2006.

4. Нешумова К. А. Электронные вычислительные машины и системы. – М.: «Высшая школа», 1989.

5. Лысиков Б. Г. Арифметические и логические основы цифровых автоматов. – М.: «Высшая школа», 1980.

6. Информатика. Учебник / Под ред. проф. Н.В. Макаровой. — М.: «Финансы и статистика», 1997.

7. Черняк Н. Г. Архитектура вычислительных систем и сетей. — М.: «Финансы и статистика», 1986.

8. Савельев А. Я. Прикладная теория цифровых автоматов. – М.: «Высшая школа», 1990.

9. Tanenbaum A.S. Structured Computer Organization (5th Edition). – UPPER SADDLE RIVER, NEW JERSEY 07458.

10. Аблязов Р.З. Программирование на ассемблере на платформе x86-64. —М.: «ДМК Пресс», 2011.

11. Рудольф Марек. Ассемблер на примерах. Базовый курс. СПб.: «Наука и техника», 2005.

12. Юров В.И. Ассемблер. Учебник для ВУЗов. СПб., 2003.

13. http://osinavi.ru/index.php?param2=18

MUNDARIJA

4-bob. KETMA-KET RAQAMLI QURILMALARNING SXEMOTEXNIKASI

5-bob. MIKROPROTSESSOR ARXITEKTURASI

6-bob. HISOBLASH TIZIMLARIDA XOTIRANI TASHKIL ETISH STRUKTURASI

7-bob. ASSEMBLER TILI

U13 Ubaydullayeva Sh.R.

Hisoblash va mikroprotsessorli texnika asoslari. Kasb-hunar kollejlari uchun o‘quv qo‘llanma (Qayta ishlangan va to‘ldirilgan 4-nashri) /Sh.R. Ubaydullayeva, K.Z. Abidov, Z.K. Abidova; O‘zbekiston Respublikasi Oliy va o‘rta maxsus ta’lim vazirligi; O‘rta maxsus, kasb-hunar ta’limi markazi. — T.: «ILM ZIYO», 2016. — 152 b.

UO‘K: 007(075)
KBK 32.973.26-04

ISBN 978-9943-16-355-3

SH.R. UBAYDULLAYEVA, K.Z. ABIDOV, Z.K. ABIDOVA

HISOBLASH VA MIKROPROTSESSORLI TEXNIKA ASOSLARI

Kasb-hunar kollejlari uchun o‘quv qo‘llanma

Qayta ishlangan va to‘ldirilgan 4-nashri

Toshkent — «ILM ZIYO» — 2016

Muharrir *N. G‘oipov*
Badiiy muharrir *M. Burxonov*
Texnik muharrir *F. Samadov*
Musahhih *T. Mirzayev*

Noshirlik litsenziyasi AI № 275, 15.07.2015-y.
2016-yil 25-avgustda chop etishga ruxsat etildi. Bichimi 60×90 $^{1}/_{16}$.
«Times» shriftida terildi. Nashr tabog‘i 8,0. Bosma tabog‘i 9,0.
1047 nusxa. Buyurtma № 64

«ILM ZIYO» nashriyot uyi, Toshkent, Navoiy ko‘chasi, 30-uy.
Shartnoma № 29 — 2016

«PAPER MAX» xususiy korxonasida chop etildi.
Toshkent, Navoiy ko‘chasi, 30-uy.