ЎЗБЕКИСТОН РЕСПУБЛИКАСИ
ОЛИЙ ВА ЎРТА МАХСУС ТАЪЛИМ ВАЗИРЛИГИ

ГУЛИСТОН ДАВЛАТ УНИВЕРСИТЕТИ

ТИЗИМЛИ ПРОГРАММАЛАШ

фанидан маърузулар тўплами

Гулистон — 2009

Ушбу маърузалар тўплами амалдаги дастурлар асосида тайёрланган бўлиб, олий ўқув юртларининг 5480100-**Амалий математика ва информатика** бакалавр таълим йўналишида таҳсил олаётган талабалар учун мўлжалланган. Унда замонавий педагогик технологиялар тизимига асосланган ҳолда назарий материаллар ва билимларни назорат қилиш учун саволлар мажмуаси кабилар келтирилган.

Маърузалар тўплами Гулистон давлат университети Ўқув – методик Кенгаши томонидан («___»_____2009 й. даги, №__сонли баённома) нашрга тавсия этилган.

Тузувчи: ўқит. Негматуллаев З.Т.

Тақризчилар: И.И. Исоқов, “Амалий математика ва информатика” кафедраси мудири.

Д.Б. Абдурахимов, “Амалий математика ва информатика” кафедраси доценти.

© Гул ДУ, 2009

Сўз боши

Ушбу тақдим этилаётган маърузалар тўплами амалий математика ва ахборот технологиялари йўналиши бакалавриятининг 3-босқич талабалари учун ”Тизимли программалаш” фанидан маъруза матнларини ўз ичига олади.

Маърузалар тўплами тизимли программа таъминотининг асосини ташкил қилада ва унинг ЭХМ архитектураси билан ўзаро боғлиқлик тамонлари кўрилади. Жумладан, ассемблер жараёни, обеъкт кодни ҳосил қилишни алгоритм ва жадваллари, обеъкт программа тузилиши ва кўчувчи программаларни яратиш, юклагичлар, макро-кенгайтмалар, компилияторларни ишлаш принциплари ёритилган, операцион тизимлар тузилиши ва вазифалари, MS DOS ва WINDOWS операцион тизимлари ҳақида асосий тушунчалар берилган.

Тизимли программалаш фанининг асосий мақсади компьютернинг программа таъминоти билан боғлиқ тушунчаларни ва ахборот технологияларидан фойдаланиш, маълумотларни қайта ишлаш масалаларини ечиш ҳақидаги маълумотлар тўпламини беришдан иборат. Курснинг асосий вазифаси – бўлажак мутахассисларга замонавий ахборот технологияларидан фойдаланиш, программа маҳсулотларини яратишдаги талаб қилинадиган компьютернинг программа ва аппарат таъминотини ўрганишларига замин яратишдан иборат.

Талабаларда фанни ўрганиш натижасида шахсий компьютерларда малакали ишлаш сабоқларини эгаллаш, информатика ва дастурлаш, компьютерда амалиёт, операцион тизимлар, тармоқ технологиялари каби курслардан эгаллаган билимларини янада мустаҳкамлашдан иборат бўлиб, бунда олий алгебра, дискрет математика, информатика фанларидан бошлнғич билим ва малакаларни талаб этилади. Фанни ўрганишда информатика ва ахборот технологиялари, дастурлаш асослари, компьютерда амалиёт, операцион тизимлар каби фанларидан олинган билим ва кўникмалардан фойдаланилади.

Таянч ибора ва тушунчалар:

1. Тизимли программалаш (ТП).
2. Соддалаштирилган ўқув машинаси (СЎМ).
3. Адреслаш усуллари (АУ).
4. Ассемблер жараёни(АЖ).
5. Обеъкт программа тузилиши (ОПТ).
6. Буйруқ ва берилганлар формати (ББФ).
7. Программаларни кўчириш (Пк).
8. Бошқарув секцияси (Бс).
9. Ташқи номлар аниқловчиси (ТНА).
10. Макро кўрсатмалар (МК).
11. Макропроцессор (МП).
12. Автомат юклагич (АЮ).
13. Компляция босқичлари (Кб).
14. Лексема (Лма)
15. Грамматика (Гка).
16. Код генерацияси процедуралари (КГП).
17. Операцион тизим вазифалари (ОТВ).
18. Операцион тизим турлари (ОТТ).
19. Мултипрограммали операцион тизимлар (МПОТ).
20. Супервизор.
21. Ўқиш, ёзиш буйруғи.
22. Реал хотира тақсимоти (РХТ).
23. Операцион тизим сиғими (ОТС).
24. Вертуал хотира (РХ).
25. Файл тизими (ФТ).
26. Операцион тизимнинг шажаравий тузилиши.
27. Операцион тизимнинг ўзаги.
28. Операцион тизимнинг қатламлари.
29. Вертуал режим.
30. Windows архитектураси.
31. Диспетчер.
32. Хотира тақсимоти.
33. Оқимларни бошқариш.

1-мавзу. **Тизимли программалашга кириш**

Асосий саволлар:

1. *Тизимли ва тадбиқий программаларни фарқи.*
2. *Соддалаштирилган укув машинаси.*
3. *Операнд адресини ҳисоблашда X регистридан фойдаланиш.*
4. *Қўшимча воситали соддалаштирилган ўқув машинаси*

Таянч ибора ва тушунчалар: *Соддалаштирилган ўқув машинаси, регистрлар, берилганлар формати, адреслаш, ўқиш-ёзиш воситалари.*

Ушбу мавзу остида кўриладиган асосий муаммо –тизимли программа таъминотининг (ТПТ) айрим ташкил қилувчиларини (компоненталарини) лойиҳалашдир. Бунда ТПТ ва ЭҲМ архитектураси ўртасидаги боғланишлар ўрганилади ва ассемблерлар, юклагичлар, макропроцессорлар, компиляторлар ва операцион тизимлар кўриб чиқилади.

ТПТ ва тадбиқий программа таъминоти бир-биридан фарқлайдиган асосий хусусиятлардан бири машинавий боғлиқликдир. Одатда тадбиқий программа бизни айрим тадбиқий масалаларни ечиш нуқтаи назаридан қизиқтиради ва унда ЭҲМ восита сифатида қаралиб, асосий эътибор масалани моҳиятига қаратилади. ТПТ эса ЭҲМ функцияларни бошқариш учун яратилади ва у бирорта конкрет масалани ечишга қаратилмайди, шу сабабли ТПТ машина тузилишига бевосита боғлиқ бўлиб қолади. *Мисол учун,* ассемблерлар мнемоник кўрсатмаларни таржима қилишда машина буйруқлар формати, адреслаш усуллари ва шунга ўхшаш хусусиятларни инобатга олиш керак.

Шу билан биргаликда ТПТ ЭҲМ турига боғлиқ бўлмаган тамонлари ҳам борки, уларда умумий схема ва ассемблерлар алгоритмлари кўпгина ЭҲМ лар учун фарқ қилмайди. *Мисол учун*, компилятор тамонидан ишлатиладиган объект кодини оптималлаштириш усуллари.

ТПТнинг умумийлик ва алоҳидалик хусусиятларидан келиб чиққан ҳолда, олдимизга қўйилган муаммони ечишда қуйидагича йўл танланади: умумийлик хусусиятларини ечиш учун реал ЭҲМларда жуда кўп учрайдиган аппарат имкониятларини ўзига олган гипотетик (мавхум) ЭҲМ **— Соддалаштирилган укув машинаси (СЎМ),** фойдаланиш ва алоҳидаликни кўрсатиш учун конкрет ЭҲМларидаги (жумладан, IBM туридаги шахсий компьютер) ТПТ компоненталарини амалга ошириш қаралади.

Соддалаштирилган ўқув машинаси. Кўп хусусиятлари бўйича соддалаштирилган ўқув машиналар (СЎМ) микро ЭҲМларга ўхшашдир. СЎМ икки вариантда бўлиб, биринчиси стандарт модел, иккинчиси СЎМ/ҚВ (ҚВ–қўшимча воситали). Бу икки вариантда “пастдан-юқорига” мослашув мавжуд бўлиб, СЎМ да яратилган объект программа СЎМ/ҚВ да ҳам тўғри бажарилади.

Соддалаштирилга ўкув машинасининг тузилиши

Хотира. Оператив хотира 8-разрядли байтлардан иборат. Учта кетма-кет жойлашган байтлар сўзни ташкил қилади. СЎМ байтли адресланган сўз –энг кичик номерли байт билан номерланади. Оператив хотира ўлчами 32768 (2^15) байтдан иборат.

Регистрлар. Бешта регистр мавжуд бўлиб, уларнинг хар биттасини ўз вазифаси бор. Қуйидаги жадвалда бу регистрлар ҳақида маълумот берилган.

Мнемоник номи	Номери	Вазифаси.
A	0	Сумматор. Арифметик амалларни бажаришда Ишлатилади.
X	1	Индекс регистри. Адреслашда ишлатилади.
L	2	Боғловчи регистр. қисм программага ўтишда (JSUB)бу

Pc	8	регистрда қайтиш адреси сақланади. Буйруқлар санагичи. Бу регистр навбатдаги бажарилувчи буйруқ адресини сақлайди.
Sw	9	Ҳолат сўзи. Бу регистр тизим маълумотларни сақлайди.Унда шарт коди (Cc-Conditional code) ҳам киради.

Берилганлар формати. Бутун турдаги қийматлар 24-разрядли иккилик сон кўринишда сақланади. Белгилар 8-разрядли ASCII кодида сақланади. Сузувчи нуқтали ҳақиқий сонлар устида амллар СЎМ да кўзда тутилмаган.

Буйруқлар формати. СЎМ барча машина буйруқлари 24-разрядли форматда бўлади.

8	1	15
Операция коди (op)	X	Адрес (addr)

Бу ерда **X** белгиси индексли адреслаш берилганда ишлатилади.

Адреслаш усуллари. Икки хил адреслаш усули мавжуд бўлиб у **X** разряд билан аниқланади. Мақсад адресни (Target address) топиш қоидаси қуйидагича.

Адреслаш усули.	**Адреслаш аломати.**	**Мақсад адресни ҳисоблаш**
Тўғридан-тўғри	X=0	TA = addr
Индексли	X=1	TA =addr+(x)

Бу ерда (x) ёзуви X-регистрдаги қийматни англатади.

Буйруқлар тизими. Соддалаштирилган ўкув машинаси буйруқлар тизимини қуйидагилар ташкил этади:

LDA, LDX — регистрларга (A,X регистрларга) қиймат юклаш буйруқлари.

STA, STX — регистрлардаги қийматларни хотирага юклаш буйруқлари.

ADD,SUB,MUL,DIV— бутун сонли арифметик буйруқлари.

COMP — сумматордаги қийматни оператив хотирадаги сўз қиймат билан солиштириш буйруғи. Бу буйруқ шарт СС кодини аниқлайди ва унда текшириш <,=,> натижаларини акс эттиради.Ўтиш буйруқлари— JLT(<), JEQ (=), JGT(>) СС шарт кодини текшириб бошқарувни керакли адресга узатилади.

JSUB, RSUB — қисм –программага ўтиш ва асосий программага қайтиш буйруқлари. Бунда L регистрдан қайтиш адресини сақлашда ишлатилади.

Ўқиш– ёзиш воситалари. Стандарт Соддалаштирилган ўкув машинаси (СЎМ) да ўқиш ва ёзиш байтларда амалга оширилади. Ҳар бир ташқи қурилма 8- разрядли код билан белгиланади. Ўқиш – ёзиш учун учта буйруқ аниқланган: TD қурилма ҳолатини текширувчи буйруқ, қурилма навбатдаги байтни қабул қилиш ёки чиқаришга тайёр ёки йўқлигини аниқлайди. Текшириш натижаси СС шарт кодида ҳосил бўлади. Агар шарт коди “=” бўлса, қурилма банд,”<”– қурилма тайёр,”>” — қурилма носоз ёки машинага уланган эмас. Бунда программа қурилма тайёр бўлгунча кутиб туриш керак; RD,WD байтни ўқиш ёки ёзиш (чиқариш). Ҳар бир байт учун RD,WD алоҳида бажарилиши керак.

Қўшимча воситали соддалаштирлган ўқув машинаси нинг тузилиши

Хотира. Хотира ҳажми 1Мбайт (2^20байт) Хотиранинг бундай кенгайганлиги буйруқлар форматини ва адреслаш усулларини ўзгартиришни тақоза қилади.

Регистрлар. СЎМ/ҚВдаги қўшимча регистрлар.

Мнемоник номи	Номери	Вазифаси.
B	3	База регистри. Адреслаш учун ишлатилади.
S	4	Умумий ишчи регистр.Махсус вазифаси йўқ.
T	5	Умумий ишчи регистр.Махсус вазифаси йўқ.
F	6	Сузувчи нуқтали сумматор(48 разрчдли)

Берилганлар формати. қўшимча равишда сузувчи нуқтали берилганлар формати.

I	**II**	**36**
S	Тартиби	Мантиса

Мантиса 0 ва 1оралиғидаги сон деб ҳисобланади , яъни нуқта унинг катта разрядли олдида жойлашган деб қаралади. Нормаллашган сон учун мантиса катта разряди бирга тенг бўлиши керак.Тартиб ишорасиз иккилик сон бўлиб, у [0..2047] оралиғида бўлади. Агар соннинг тартиби е –сонига, мантиса f сонига тенг деб ҳисобласак, бу форматда соннинг абсолют қиймати fx2 ^ (e-1024) бўлади.Сон ишораси S разряд билан аниқланади. Машина ноли барча разрядларда нол бўлган сўз билан тасвирланади.

Буйруқлар формати. СЎМ/ҚВда катта хажмда хотира учун СЎМдаги 15 разрядли хотира майдони етарли эмас. Бу муаммони ечишнинг икки хил усули бор: биринчиси – нисбий адреслашдан фойдаланиш, иккинчиси адрес майдонини кенгайтириш. СЎМ/ҚВда бу усулларнинг иккаласидан ҳам фойдаланилади.

СЎМ/ҚВдаги буйруқ форматлари:

1. 1- байтлик 8

Операция коди (Op)

2.2-байтлик 8 4 4

Операция коди (Op)	R1	R2

3.3-байтлик

6	1	1	1	1	1	1	12
Опер.коди	n	i	X	b	p	e	Силжиш(disp)

4.4-байтлик

6	1	1	1	1	1	1	20
Опер.коди	n	i	X	b	p	e	Адрес (addr)

Адреслаш усуллари. Стандарт СЎМга нисбатан СЎМ/ҚВда нисбий адреслашнинг иккита усули аниқланган (3-формат бўйича).

Адреслаш усули	Адреслаш аломати	Мақсад адресни ҳисоблаш ҳисоблаш
Базага нисбатан	b=1, p= 0	ТА к(B)+ disp, (0≤disp≤4095)
Буйруқ ҳисоблагичига нисбатан	b =0, p=1	ТАқ(Pc)+disp, (-2048≤disp≤2047)

База бўйича адреслашда силжиш майдони 12 разряди ишорасиз бутун сон деб қаралади. Буйруқлар ҳисоблагичига нисбатан адреслашда 12 разрядли силжиш ишорали бутун сон деб қаралади. Агар 3- формат буйруғида b=0, p=0 бўлса, мақсад адреси disp майдонидан олинади. 4-формат буйруғида b=0, p=0 бўлса, мақсад адрес addr майдонидан

олинади. Адреслашнинг бу ҳолига тўғридан –тўғри адреслаш дейилади. Юқоридан келтирилган адреслашлар индексли адреслаш билан биргаликда олиб борилиши мумкин, бунда х-разряд қиймати 1 ва адрес ҳисобига (X) қиймати қўшилади. 3-4- формат буйруқларида i ва n разрядлар мақсад адресни ишлатиш усулини билдиради.

Агар i=1 ва n=0 бўлса, мақсад адрес қиймати операнд сифатида ишлатилади. Бу усулга *бевосита адреслаш* дейилади.

Агар i=0 ва n=1 бўлса, мақсад адрес бўйича сўз қиймати операнд адреси сифатида ишлатилади. Бу усулга *воситали адреслаш* дейилади.

Агар бир вақтда i ва n бир вақтда 0 ёки 1 бўлса,мақсад адрес қиймати операнд жойлашиш ўрнини беради. Бу усулга *оддий адреслаш* дейилади.

Буйруқлар тизими. Стандарт СЎМга қўшимча равишда LDB, STB-B регистрга ёзиш ва B регистрдан хотирага ёзиш. ADDF, SUBF, MULF, DIVF — сузувчи нуқтали арифметик амаллар. RMO —бир регистрдан иккинчи регистрга узатиш. ADDR, SUBR, MULR, DIVR — регистр қиймати устида амал бажариш. SVC — супервизорга мурожаат қилиш. қуйида СЎМ/ҚВда адреслашга доир мисол келтирилган.

А регистрга қиймат юклаш буйруғи (LDA) бажарилиши, буйруқ коди 00H.

…..	
3030	003600
…..	
3600	103000
6390	00C303
……	
C303	003030

(B) = 006000
(Pc) = 003000
(X) = 000090

Ушбу мисол адреслаш усулларини чуқурроқ тушиниб олишга ёрдам беради.

Назорат саволлари

1. Тизимли ва тадбиқий программалар фарқи нимада?
2. Нима учун СЎМда оператив хотира ўлчами 32768 байтдан катта бўла олмайди.
3. Операнд адресини ҳисоблашда X регистрдан фойдаланишга зарурат нимада?
4. СЎМда ҳар бир ўқиш-ёзиш амалида берилганлар ўлчами (узунлиги) неча байт?
5. СУМ/ҚВда оператив хотира 1Мб кенгайганлигини тушинтириб беринг?
6. 25,6 сонини сузувчи нуқтали берилганлар форматида тасвирлаб беринг?
7. Нима мақсадда СЎМ/ҚВда буйруқлар формати 4 хил қилиб аниқланган?
8. СЎМ/ ҚВда 3 ва 4 формат буйрукларида адреслашнинг қандай усуллари қулланилган?
9. Юқорида келтирилган мисолдаги жадвалнинг 3-сатрида А регистрга 103000 қиймат юкланишини тушунтириб беринг?

2-мавзу. **Ассемблер жадвали ва алгоритми**

Асосий саволлар

1. *Ассемблер тили ва унинг вазифалари.*
2. *Ассемблер тили буйруқлари.*
3. *Ассемблерда псевдобуйруқлар.*
4. *Берилганларни аниқлаш.*

Таянч ибора ва тушунчалар: мнемокодлар, объект программа, юклагич, операндалр, псевдобуйруқлар, идентификаторлар.

1. *Ассемблер тили ва унинг вазифалари.*

Ассемблер-бу мнемокодлардан тузилган программа. Шу тилда тилда тузилган программани *бошланғич программа* деб атаймиз. МП буйруқларини сонли кодлардан тузилган программани *объект программа* деб атаймиз. *Демак*, ассемблернинг вазифаси бошланғич программани МП тушунадиган объект программага ўтказишдан иборат. IBM фирмасининг иккита программалар пакети бор-кичик ассемблер (ASM) ва макроассемблер (MASM). ASM учун бор-йўғи 64 К керак бўлади, MASM эса 96 К талаб қилади.

Уларнинг бир-биридан фарқи:

- ASM хатоларнинг фақат кодини беради;
- MASM хатонинг ўзини ҳам чоп этади, 80287 ва 80286 МП ларнинг буйруғини ҳам тушунади, макротаърифларни ишлатади.

Юклагич. Аввало программани бирор матн муҳарририда ёзиб файл ташкил этиш керак бўлади. Энди тузилган программани ассемблерлаш керак бўлади. Бунинг учун дискдан ASM.EXE (ASM учун) ёки MASM.EXE (MASM учун) файллардан бирортасини топиб ишлатиш керак. Шунда дисплей эаркнига қуйидаги маълумотлар чиқади:

```
Source  filename    [.asm]
object filename     [filename.obj]
source listing      [nul.lst]
cross-reference     [nul.crf].
```

Шунда курсор биринчи қаторнинг охирида туради, унга бошланғич программа номини киритиш керак бўлади. Файл кенгайтмаси .ASM ни ёзиш шарт эмас, чунки ассемблер ўзи уни турибди деб фараз қилади. Иккинчи қатордаги сўровга ҳосил қилинадиган объект программасининг номи, керак бўлса керакли дискюритувчи номини ҳам кўрсатиш мумкин. Учинчи қатордаги сўров ошкормас равишда программани ассемблерлаш керак эмаслигини қабул қилади. Агарда листинги керак бўлса дискюритувчини номини кўрсатиш керак. Агарда ошкормас қийматларни қабул қилиш керак деб ҳисобласангиз, у ҳолда охириг учта сўровга <ENTER> клавиши босиб жавоб беришингиз мумкин.

Энди ҳосил қилинган объект программадан бажариладиган программа ҳосил қилиш учун юклагич программа LINK.EXE ишлатилади. Бу программа ишлатилганда дисплей экранига қуйидаги маълумотлар чиқади:

```
Object   Modules     [.obj]
         Run  File   [номи.exe]
         List  File  [nul. map]
         Libraries   [.lib]
```

Биринчи сўровга объект программанинг исми берилади, бунда .obj файл кенгайтмасини бермаса ҳам бўлади.Иккинчи сўровга бажарилувчи файл номини бериш керак бўлдаи. Керак бўлса дискюритувчи номини кўрсатиш мумкин. Учинчи сўровга CON жавобини киритиш керак, акс ҳолда кесишмалар жадвали MAP файл дискда ҳосил

бўлади ва бекорга жой олади. CON жавобини берганда эса, у файл дисплей экранида ҳосил бўлади. Тўртинчи сўровга <ENTER> ни босиш керак, шунда LINK юклагич қолган параметрларни ошкормас равишда қабул қилади.

DEBUG созловчиси. **DEBUG** созловчи программанинг ҳар бир буйруғини кетма-кет бажарилишини кўрсатиб туради. **DEBUG** нинг қуйидагича буйруқлари мавжуд:

G n – n адресли буйруқгача программани бажариш;
Dn - хотиранинг қийматини ички (ўн олтилик) кўринишда n-адресдан бошлаб чиқариш;
U n – хотиранинг қийматини ташқи (белгили) кўринишда n-адресдан бошлаб чиқариш;
R - Регистрнинг қийматини чиқариш;
Q - DEBUG дан чиқиш.

Бу буйруқда кўрсатилган n икки қисмдан иборат: x : y. Бу ерда x-регистр номи, y- кўчиш.

Программани ишга тушириш. Энди ҳосил бўлган .EXE файлли программани бажаришга юбориш мумкин.

2. Ассемблер тили буйруқлари

Оператор. Ҳар қандай программа ҳам операторлар кетма-кетлигидан иборат бўлади. Ассемблерда оператор ёки буйруқ ёки псевдобуйруқ (директива) кўринишида бўлиш мумкин. Ихтиёрий оператор амалларни ва улар орқали боғланган операндларни ўз ичига олади. Улар ўз навбатида ифодаларда иштирок этади. Ифодада белгилар сатри ёки ўзгармас катталик (константа) қатнашиши мумкин. Белгилар сатри қўштирноқлар ичига олиб ёзилади. *Масалан*: ‘PC’ ёки “PC”. Ассемблер уларни ASCII форматдаги объект кодига ўтказади. Сонли ўзгармас катталикни ва хотира адресини ифодалашда ишлатилади. Ассемблер ҳамма сонли константаларни *ўнолтиликка* ўтказади ва ўнгдан чапга қараб ёзиб объект кодларида байтларга ёзади.

Буйруқ формати. Ассемблер тилининг ҳар бир буйруғи қуйидаги майдонлардан иборат бўлиши мумкин:

[нишон:] мнемокод [операнд] [;изоҳ].

Бу ерда буйруқда мнемакод турган майдонни бўлиши албатта шарт, қолганларидан операнд майдони керак бўлганда ишлатилади. Майдонлар ўртасида камида битта пробел (бўш жой) бўлиши керак.

Нишонлар майдони. Бу майдон ассемблер тили буйруғига исм бериш учун хизмат қилади. Бу вазифа билан нишонлар Бейсик тилидаги оператор олдида турадиган сатр номери ишни бажаради. Буйруқ нишони 31 тагача бўлган белгилар ва рақамлар кетма-кетлигидан иборат бўлиб, биринчиси албатта ҳарф бўлиши керак. Нишон «:» белгиси билан тугайди. Нишон сифатида ишлатиладиган кетма-кетликда қуйидагилар ишлатилиши мумкин:

-A дан Z ёки a дан z гача бўлган ҳарфлар;
-0 дан 9 гача бўлган раҳамлар;
-? @ _ $ белгилари.

Нишон сифатида регистрлар номлари, буйруқ мнемакодаларини ишлатиш мумкин эмас.

Мисоллар.
```
GET_Count:  MOV  CX, DI
A25:        MOV  DEST, 25H
```

Мнемокод майдони. Мнемакод майдони МП буйруғининг исмини ўзида сақлайди. Исмлар 2 тадан 6 тагача ҳарфлардан иборат бўлиши мумкин. *Масалан,* ADD- қўшиш буйруғининг исми, MOV-берилганларни кўчириш буйруғининг исми, LOOPNZ-ноль бўлмагунча такрорлаш буйруғининг исми. Мнемакод бўйича ассемблер нечта операнд ва улар қайси турга эгалигини аниқлайди.

Операндлар майдони. Агар буйруқ мнемакод бажарилиши керак бўлган ҳаракат амални ифодаласа, операнд МП га қайта ишлаш учун зарур бўлган берилганни қаердан қидириш кераклигини кўрсатади.

Масалан: MOV CX, DX.

Бу буйруқда кўрсатилган DX регистрида турган нарса (катталик) CX регистрига кўчирилиши керак. Буйруқ умуман олганда операндсиз, битта опернадли ёки икки опернадли бўлиши мумкин. Операндлар бир-бири билан вергул билан ажратилади.

```
Масалан:    RET                 ; қайтариш.
            INC  CX             ; CX ни ошириш
            ADD  AX,12          ; 12 ни AX га қўшиш.
```

Икки операндли буйруқлардаги биринчи операнд **қабул қилувчи,** иккинчиси **узатувчи (манба)** дейилади.

Изоҳлар майдони. Изоҳ программани ўқилишини қулайлаштирш учун хизмат қилади. Изоҳдан олдин нуқта вергул (;) белгиси қўйилади. Изоҳ айрим ҳолда якка ўзи келиб бутун сатрни эгаллаши мумкин.

```
Масалан:  MOV CX, 0    ; CX регистрига 0 ни юклаш.
          ; Ассемблер МП  8088
```

3. Ассемблерда псевдобуйруқлар

Ассемблерда шундай операторлар борки, улар ассемблер ишини бошқаради. Булар фақат ассемблерлаш жараёнида ишлатилади ва машина (объект) кодларини ишлаб чиқмайди. *Псевдобуруқлар* ёрдамида программа листингини бошқариш, сегмент ва процедураларни аниқлаш, буйруқларга ва берилганларга исм бериш, хотирани ишчи жойларини вақтинча банд этиш мумкин.

[Идентификатор] псевдобуйруқ [операнд] [изоҳ]

Бундан кўриниб турибдики, псевдобуйруқ баъзи бир опернадларни ишлатмайди, баъзилари бўлса ишлатади.

Листингни бошқарувчи псевдобуйруқлар. Листингни бошқарувчи учта псевдобуйруқ бор: PAGE, TITLE, SUBTIL..

PAGE псевдобуйруғи ёрдамида листингни бир бетида чиқарилиши мумкин бўлган сатрлар ва ҳар бир сатрдаги белгилар сонини бошқариш мумкин:

PAGE [сатрлар сони] [, белгилар сони]

Ҳар бир бетдаги сатрлар сони 10 тадан 255 тагача, сатрдаги белгилар сони 60 тадан 132 гача. Ошкормас равишда *PAGE* 66, 60 қабул қилинган. Агарда операндларсиз *PAGE* псевдобуйруғи ишлатилса, ассемблер жой ташлаб янги бетга ўтади.

TITLE псевдобуйруғи листингни ҳар бир ьетини юқорисида программа боши (титулни) нинг номини чоп қилишда ишлатилади.

Масалан : TITLE PRM1

TITLE псевдобуйруғи ҳар бир бетнинг иккинчи сатрига срлавҳани чоп қилади. Сарлавҳа чапга тўғирланади.

SUBTIL псевдобурйғи ҳар бир бетнинг учинчи сатрига сарлавҳани чоп қилиш имконини беради. *Масалан : TITLE Галлактика SUBTIL Ер сегменти берилганлари.*

Битта сегментни чоп қилиб бўлгандан сўнг сарлавҳани қуйидагича ўзгартириш мумкин: *SUBTIL* Плутон сенменти берилганлари PAGE. Сарлавҳа сарлавҳа остилар 60 тагача белгилардан иборат бўлиши мумкин.

4. Берилганларни аниқлаш

Ўзгарувчилар учун берилганларни аниқлаш учун қуйидаги псевдобуйруқлар ишлатилади:

-DB байтни аниқлаш;

-DW сўзни аниқлаш;
-DD иккиланган сўзни аниқлаш;
-DQ тўртта сўздан иборат майдонни аниқлаш;
-DT ўнта байтни аниқлаш.

Умуман олганда берилганларни аниқлаш псевдобуйруқининг формати қуйидагича :

[Исми] Dn ифода [,…….]

Бу ерда n-B,W,D,Q,T лардан бирортаси. Ифода сонли ўзгармас, белгилар сатри ёки ? лардан бирортаси бўлиши мумкин.

Масалн: BU_MAX DB 255; байтнинг ишорасиз максимал қиймати;
BS_MAX DB 127; байтнинг ишорали максимал қиймати;
BS_MIN DB-128; байтни ишорали минимал қиймати;

Массив жойлаштириш қуйидагича амалга оширилади:

A_TABLE DB 0,1,2,3,-4
DB 100,0, 55, 63, 70

Агарда рўйхатда бир хил қийматлар бўлса, уларни қайтариш (такрорлаш)ни DUP амали орқали ёзса бўлади:

Масалан: A, DB 0,0,0,1 ни қуйидагича ёзса бўлади:
A DB 4 DUP(0), 1

Юқорида келтирилгандек берилгнлар аниқланганда, кўрсатилган исмларга бошланғич қийматлар берилади. Энди агарда олинадиган натижа учун жой банд қилиш керак бўлса, қуйидагича иш тутилади:

RESULT DB ?
MAC DW 12 DUP (?)

Бунда *RESULT* исмли ўзгарувчига хотирадан бир байт жой банд қилинган бўлса, *MAC* номли массив учун хотирадан 12 та сўз жой банд қилинди. Псевдобуйруқлар ёрдамида яна хотира ячейкалари адресларини сақлашда ҳам фойдаланиш мумкин. *Масалдан:* PAAA DW AAA AAA. Бунинг натижасида 16 битли кўчиш меткага PAAA ном берилади. Одатда кўчишни сақлаб турувчи ўзгарувчигс *кўрсаткич* деб аталади. Бундай кўрсатгични нишон ва унга мурожаат қилаётган буйруқ битта сегментда жойлашганда ишлатилади. Агарда нишон ва унга мурожаат қилаётган буйруқ ҳар хил сегментларда жойлашган бўлса, у ҳолда нишонни кўчишидан ташқари блок номерини ҳам билиш керак. Бундай ҳолларда қуйидагича аниқланади: *PAAA DB AAA*

Адреснинг иккита компоненти (блок номери ва силжиш) ни сақлаб турувчи ўзгарувчини *вектор* деб атаймиз.

Қўшимча псевдобуйруқлар. GROUP (гуруҳ) псевдобуйруғи. Бу псевдобуйруқ хотиранинг 64К ҳажмли бир блокига бир неча сегментни гуруҳлаб жойлайди. Одатда бу псевдобуйруқ операцион системасининг буйруғи бўлмиш COM туридаги бир блокдан иборат бўладиган программани яратишда ишлатилади. *Масалан,* агарда сизнинг программангизда буйруқлар сегменти CODSEG исмга, берилганлар сегменти DATASEG исмга, у ҳолда уларни GROUP исмли, 64К жойни эгалловчи бир блокка қуйидагича гуруҳлаш мумкин:

```
        GROUP GROUP
        CODSEG, DATASEG
DATASEG  SEGMENT  PARA  PUBLIC  'DATA'
        ……….
DATASEG  ENDS
    CODSEG  SEGMENT  PARA  RUBLIC  'CODE'
        ASSUME  CS: CROUP, DS: GROUP……
        CODSEG  ENDS
            END
```

Шартли псевдобуйруқлар. Шартли псевдобуйруқлар ассемблерни трансляция вақтида бирон бир шартни “рост” ёки “ёлғон” лигига қараб берилган бир гуруҳ бошланғич операторларни ёки трансляция қилиш ёки таншлаб кетишга мажбур қилади. Шартли трансляцияни ташкил этиш учун программа матнини керакли қисми IF ва ENDIF оралиғига олинади. Агарда **IF** псевдобуйруғидаги шарт “рост” бўлса, у ҳолда IF ва ENDIF оралиғига олинган программа бўлаги трансляция қилинади, акс ҳолда ташлаб кетилади. IF псевдобуйруқлари жами сони 8 та бўлиб, уларни тўртта- тўртта қилиб икки гуруҳга қуйидагича бўлиш мумкин: Ифоданинг қиймати =0 бўлса “рост” қийматни беради; IF ифода - Ифоданинг қиймати <>0 бўлса «рост» қийматни беради;

IF1- Ассемблер биринчи ўтишни бажараётган бўлса “рост” қийматини беради;

IF2 –Ассемблер иккинчи ўтишни бажарётган бўлса “рост” қийматини беради;

IFDEF идентификатор.

Агарда идентификатор аниқланган бўлса ёки EXTERN псевдобуйруғида эълон қилинган бўлса «рост» қийматини беради;

IFDEF <1-сатр>, <2-сатр>

Агарда 1-сатр ва 2-сатрлар бир хил бўлса “рост” қийматни беради;

IFDEF <1-сатр>, <2-сатр>

Агарда 1-сатр ва 2- сатрлар бир-биридан фарқ қилса “рост” қийматни беради;

Охирги иккита псевдобуйруқдаги операндлар майдонида <> белгиларнинг бўлиши шарт. Баъзи ҳолларда программада псевдобуйруқларни ишлатиш жараёнида агарда шарт натижаси “ёлғон” бўлганда альтернатив формати ELSE дан фойдаланиш ҳам яхши натижаларни беради. Бу ҳолда шартли псевдобуйруқларнинг умумий шаклда қуйидагича ифодалаш мумкин:

```
If xx [аргумент]
……..
…… (шартнинг “рост” қийматига мос келувчи операторлар)
[ELSE]
……..
…… (шартнинг “ёлғон” қийматига мос келувчи операторлар)
ENDIF
```

Назорат саволлари

1. Нима учун Ассемблер тилида тузилган программани бошланғич программа деб атаймиз?
2. Тузилган программа қандай ассемблерга ўтказилади?
3. Ассемблер тилининг буйруқларини берилиш формасини айтинг.
4. Операндлар майдони нима ?
5. Ассемблер тилидаги псевдобурйқлар ҳақида маълумот беринг.

3-мавзу. **Буйруқлар формати ва адреслаш усуллари.**

Асосий саволлар:

1. *Трансляцияни бошқариш псевдобуруқлари.*
2. *Ассемблерда арифметик амаллар.*
3. *Умумий вазифадаги буйруқлар.*
4. *Киритиш-чиқариш буйруқлари.*
5. *Процессор томонидан буйруқни қайта ишлаш.*

Таянч ибора ва тушунчалар: *трансляция, листинг, регистр-регистр, хотира-регистр, трансляция.*

1. Трансляцияни бошқариш псевдобуруқлари. **ORG** (origin-бошлаш) псевдобурйғи. Бу псевдобуруқ дрес ҳисобчисини, ички кўрсатгични ўзгартиради, у эса ассемблерга хотиранинг қайси жойида буйруқлар ва берилганларни сақлаш кераклигини кўрсатади. Одатда бу ишларни аслида DOS операцион системаси қилади, лекин сиз ўзингиз ҳам бу ORG буйруғи ёрдамида хотирани тақсимлашингиз мумкин. *Масалан,* агарда сиз ўзингизнинг программангизни DOS лперацион системасининг .COM файли турида қилиб ишлатмоқчи бўлсангиз, у ҳолда программанинг биринчи буйруғидан олдин ORG 100H операторини қўйинг. Бу оператор ассемблерга программанинг буйруқларини хотирада сегмент буйруқларидан сўзга 256 байт жой ташлаб жойлаштириш кераклигини кўрсатади.

EVEN (жуфт) псевдобуйруғи. Одатда бу псевдобуйруғ жуда кам ишлатилади. Унинг ёрдамида 8086 ёки 80286 МПли ЭҲМларда программани эффектив бажарилишини таъминлаш мумкин. Одатда 8086 МП 16 битли берилганлар шинасига эга бўлганлиги учун бир йўла 16 бит маълумот узатиш имкониятига эга (8088 МП бунинг учун 8 битдан 2 марта узун). Лекин бу ишни бажаришда 8086 МП хотиранинг тоқ адресида жойлашган берилганларни жуфт адреслардагига қараганда узоқроқ узатади. Шунинг учун вақт жуда тиғиз бўладиган программаларда берилганларни жуфт номерли адресларда эслаш имкониятига эга бўлиш муҳим аҳамият касб этади.

2. Ассемблерда арифметик амаллар

Ассемблер тилида арифметик амаллар сонли операндлар устида бажарилади ва сонли натижа беради. Энг кўп ишлатилувчи арифметик амаллар:

қ қўшиш, - айириш, * кўпайтириш, / бўлиш. Уларнинг умумий шакли { - қ * / }. Бу ерда / амали ишлатилганда натижанинг бутун қисми олинади.

Масалан: PI_QUOT EQU 31416 / 1000 оператории 3 қийматни беради.

Ассемблерда қолдиқни ҳисобга олиш учун MOD амали мавжуд.

Масалан: PI_REM EQU, 31416 MOD 1000 оператори қиймати 1416 га тенг бўлган PI_REM ўзгармасни аниқлайди.

SHL- Чапга суриш амали.

SHR- Ўнга суриш амали. қиймати ифоданинг қийматига тенг битларга ўнгга суради.

Мантиқий амаллар. Ассемблерда мантиқий амаллар иккилик қийматлар билан иш кўради. Лекин SHL ва SHR амаллардан фарқли мантиқий амаллар айрим битлар билан ишлайди. **AND –**мантиқий **ВА**, **OR-**мантиқий **ЁКИ** ва **XOR –**мантиқий ЁКИ нинг ИНКОРИ амаллари иккита операнд устида бажарилади. NOT- бирга тўлдириш амали битта операнд устида бажарилади.

AND, OR ва XOR амаллари натижаси қуйидагича (1-жадвал):

1-операнд	2-операнд	Натижа		
		AND	OR	XOR
0	0	0	0	0
0	0	0	1	1
1	0	0	1	1
1	1	1	1	0

8088 МП ни ассемблер тили буйруқлари ичида AND, OR ва XOR буйруқлари ҳам бор. Лекин уларнинг бир-биридан фарқи шундаки, мантиқий буйруқлар программанинг бажарилишида ишласа, мантиқий амаллар уни трансляция пайтида ишлатади.

3. Умумий вазифадаги буйруқлар.

3.1. **MOV** буйруғи. Бу буйруқ ёрдамида регистр ва хотра ячейкаси ёки иккита регистр ўртасида байт ёки сўзни узатиш мумкин. Бундан ташқари, яна тўғридан- тўғри қийматни регистр ёки хотира ячейкасига жўнатиш мумкин. Буйруқнинг кўриниши қуйидагича:

MOV қабул қилувчи, узатувчи

Мисоллар:

```
MOV  AX, TAB        ;  хотирадан регистрга жўнатиш
MOV  RESULT,AX      ;  регистрдан хотирага жҳнатиш
MOV  ES: [BX], AX   ;  сегмент регистрини алмаштириш
MOV  DS, AX         ;  16  битли регистлар ўртасида жўнатиш
MOV  BL, AL         ;  8 битли регистрлар ўртасида жўнатиш
MOV  CL, - 30       ;  ўзгармас регистрга жўнатиш
MOV  DEST, 25 H     ;  ўзгармасни хотирага жўнатиш.
```

Бу буйруқ билан қуйидаги ишларни бажариш мумкин эмас:

1. Хотиранинг бир ячекасида турган берилганни тўғридан тўғри иккинчи ячейкага жўнатиш мумкин эмас. Бунинг учун уни аввл умумий регистрга жўнатиш, сўнгра умумий регистрдан хотиранинг керакли жойига жўнатиш мумкин:

Яъни MOV AX, RES
MOV Y, AX

2. Худди биринчи банддагига ўхшаб адресланувчи операндни сегмент регистрига тўғридан-тўғри жўнатиш мумкин эмас. Бунинг учун 1-банддагига ўхшаб иш тутиш керак.

Яъни, MOV AX, DATA_SEG
MOV DS, AX

Одатда бу каби буйруқлар ASSUME буйруғига мос келади ва ассемблерга берилганлар сегменти қаерда жойлашганини кўрсатади.

3. қийматни тўғридан-тўғри бир сегмент регистридан иккинчи сегмент регистрига жўнатиш мумкин эмас. Бунинг учун умумий регистрдан фойдаланиш керак. *Масалан,* DS регистри худди ES регистри кўрсатаётган сегментни кўрсатиш керак бўлса, қуйидагича иш тутилади:

MOV AX, ES
MOV DS, AX

4. Узатиш буйруғида қабул қилувчи сифатида CS регистрини ишлатиш мумкин эмас.

3.2.**PUSH ва POP буйруқлари.** Одатда бу буйруқлар программанинг ишлаш жараёнида берилганларни (регистрларда ва хотира ячекасида жойлашган қийматларн) вақтинчалик сақлаш учун хизмат қилади. *Масалан,* фараз қилайлик, сизга баъзи бир ишларни бажариш пайтида AX регистрининг қийматини сақлаб туриш зарур бўлсин. Бунинг учун қуйидаги буйруқни ишлатамиз:

PUSH узатувчи : бизнинг мисолимизда AX регистри;

PUSH буйруғи ёрдамида регистрнинг қиймати ёки 16 битли сўз кўринишидаги хотира ячейкасининг қиймати *стек* бошига жойланади.

POP буйруғи бўлса, *стек* бошидан сўз жойни эгалловчи қийматни олиб, регистр ёки хотира ячекасига жойлашади, яъни POP қабул қилувчи; бизнинг мисолимизда AX регистри.

Мисоллар:

```
PUSH  SI;          умумий регистр ёки сегмент.
PUSH  DS;          регистрни сақлаш.
PUSH CS;           CS ни сақлаш.
PUSH  DELTA;       хотира ячейкасининг қийматини
PUSH TAB [BX] [DI]  сақлаш.
```

Шуни унитмаслик керакки, программада ҳар бир ишлатилган PUSH буйруғига мос келувчи POP буйруғи бўлиши керак.

```
Масалан:    PUSH  AX ;    AX  ни стек бошига ёзиш.
            .....     ;    AX ни ўзгартирувчи бошқа.
            .....     ;    буйруқлар кетма-кетлиги
            POP  AX   ;    стек бошидан  AX га ўқиш.
```

Стек боши деб, стек кўрсатгичи SP да адреси сақланаётган стек сегментидаги ячейка тушунилади. Стек хотиранинг кичик адреслари (0 ячейкаси) га қараб *ўсиб* боради, шунинг учун стекка биринчи жойлаштирилаётган сўз стекнинг энг катта адресли ячейкасида сақланади (ёзилади), кейингиси ундан икки байт кичигига ва х.к. ёзилади.

3.3. **XCHG алмаштириш буйруғи.** Бу буйруқ ёрдамида иккита регистр ёки регистр ва хотира ячейкаси ўртасида қийматлар ўзаро алмаштирилади. Лекин бу буйруқ сегмент регистрлари қийматларини алмаштиришда ишлатилмайди.

```
Мисоллар:  XCHG  AX, BX              ; икки регистр қийматини
           XCHG  AL, BH              ; алмаштириш  (сўз ёки байтларда)
           XCHG  WORLD_LOC, DX       ; хотира ячейкаси ва регистрларнинг
           XCHG DL, BYTE_LOC         ; қийматларини алмаштириш.
```

4. Киритиш-чиқариш буйруқлари. Киритиш- чиқариш буйруқлари системанинг периферик ташқи қурилмалари билан ўзаро алоқаси учун ишлатилади. Улар қуйидаги кўринишда бўлади:

```
IN  аккумулятор, порт
OUT порт, аккумулятор
```

Бунда аккумулятор сифатида –сўзлар билан ишланганда AX регистри, байтларда бўлса, AL регистридан фойдаланилади. Порт операнди сифатида 0 дан 255 гача бўлган ва 256 қурилмага мос келувчи, ўнлик қиймат бўлиши мумкин. Порт операнди сифатида DX регистрини ишлатиш мумкин. Бунда у порт номерини ўзгартириш имконини беради, *масалан,* керак бўлганда бир хил берилганларни ҳар хил портларга узатиш мумкин.

Мисоллар:

```
IN AL, 200              ;  200 портдан байтни ўқиш
IN AL, PORT_ VAL        ;  ўзгармас кўрсатаётган портдан байтни ўқиш
OUT  3OH, AX            ;  30H портга сўзни чиқариш
OUT  DX, AX             ;  сўзни чиқариш.
```

Қуйида СЎМ учун тузилган прграммани (1-программа) СЎМ/ҚВ учун ҳисоблаш самарадорлиги ошириш нуқтаи назаридан ўзгартирилган варианти келтирилган (2-программа).

Сатр	Адрес	Нишон	Буйруқ	Операнд	Объект коди	Изоҳ
5	0000	COPY	START	0		Файлдан нусха олиш
10	0000	FIRST	STL	RETADR	17202D	қайтиш адресини сақлаш.
12	0003		LDB	#LENGTN	69202D	
13			BASE	LENGTN		База регистрини ўрнатиш.
15	0006	CLOOR	JSUB	RDRES	4B101036	Ёзувни ўқиш.
20	000A		LDA	LENGTN	032026	Маълумот тугашини аниқлаш.
25	000D		COMB	#0	290000	
30	0010		JEQ	ENDFIL	332007	Охири бўлса,чиқиш
35	0013		JSUB	WRREC	4B10105D	Берилганларни чиқари
40	0017		J	CLOOR	3F2FEC	Такролаш учун ўтиш
45	001A	ENDFIL	LDA	EOF	032010	Файл охири белгисини юклаш
50	001D		STA	BUFFER	0F200D	
55	0020		LDA	#3	010003	
60	0023		STA	LENGTN	0F200D	LENGTN 3
65	0026		JSUB	WRREC	4B10105D	
70	002A		J	@RETADR	3E2003	Программадан чиқиш.
80	002D	EOF	BYTE	C’EOF’	454F46	
95	0030	RETADR	RESW	1		
100	0033	LENGTH	RESW	1		Ёзув узунлиги
105	0036	BUFFER	RESB	4096		Буфер узунлиги–4096 байт.

Программада СЎМ/ҚВ имкониятларидан тўла фойдаланиш мақсадида маълум бир ўзгаришлар киритилган. START қиймати 0 бўлиши бу программани кўчувчи программа эканлигини билдиради. Программада регистр – регистр, хотира – регистр форматидаги буйруқлардан кенг фойдаланилган. Регистрга мурожаат тез бажарилиши ҳисобига программа бажарилиш самарадорлиги оширилган. Энди объект код қурилишини таҳлил қилайлик. Программадаги регистр – регистр буйруқлари — CLEAR учун COMPR объект кодни кўриш бирор қийинчилик туғдирмайди (улар ОРТАВ жадвалида берилади).

Хотира – регистр буйруқлари – нисбий адреслашдан фойдаланилади (буйруқ ҳисоблагичи ёки база регистри бўйича). Ҳар қандай ҳолда ҳам ассемблер силжиш адресини ҳисоблаши керак. Нисбий адрес қандай аниқланишини кўрайлик. Ассемблер мақсад адресни ҳисоблаш учун тескари иш қилади. *Мисол учун* Рс регистрига нисбатан ҳисобланувчи буйруқ . 10 0000 FIRST STL RETADR 17202D

Процессор тамонидан STL буйруғини бажаришда Рс навбатдаги буйруқ адресини (0003) кўрсатиб туради. RETADR адреси 0030 (ассемблер бу маълумотни SYMTAB жадвалидан олади). Биз учун керак бўлган силжиш 30 –3к2D бўлади.

Программа бажарилишида Disp+(Рс) орқали керакли адрес (0030) топилади. Буйрукда р ни разряди 1 бўлади.

Мисол учун,

40 0017 J CLOOP 3F2FEC

Бу буйрукда операнд адреси 0006. Процессор тамонидан буйруқни қайта ишлаш пайтида Рсқ0001А қийматга тенг бўлади. Силжиш 6 – 1Ақ-14. Бу сонни қўшимча коди FEC кўринишида бўлади. Худди шу усулда В регистр учун силжиш топилади. Фарқи шундан иборатки, Рс регистри қиймати автоматик равишда аниқласа, В регистр қиймати программа тузувчи тамонидан берилади (BASE ва NOBASE калит сўзлари орқали бошқарилади).

Программада disp майдонига силжиш қиймати сиғмаган холлар учун 4-формат буйруғидан фойдаланилган. Бу буйруқларда операнд адреси 20 – разрядлик майдонга тўлиқ ёзилган (абсолют адрес). Бу формат буйруқни ишлатишни программа тузувчининг ўзи аниқлаши ва буйруқ олдига «+» белгисини қўйиш керак.

15 0006 CLOOP JSUB RDREC 4B101036

Бевосита операндлардан фойдаланувчи буйруқлар трансляцияда хотирага мурожаат бўлмайди (CLEAR, буйруғи).

55 0020 LDA #3 010003

Бевосита адреслашга бошқа мисол

12 003 # LENGTN 69202D

Бу буйруқ бевосита операнд LENGTN номи билан берилган ва унинг қиймати адрес бўлгани учун В регистрга LENGTN адреси юкланади.

Назорат саволлари

1. Трансляцияни бошқариш учун қандай псевдобуйруқлар мавжуд?
2. Ассемблерда арифметик амалларнинг бажарилиши.
3. Умумий вазифадаги буйруқларнинг турлари ва вазифалари.
4. Киритиш-чиқариш буйруқларининг вазифалари.
5. Процессор томонидан буйруқни қайта ишлаш.
6. СЎМ/қВда учун тузилган программада 4-формат буйруғини ишлатишга сабаб нима?
7. ” STCH BUFFER,X “ буйруғда нимани билдиради.
8. СЎМ/ҚВ учун тузилган программада операнд адреси қайси регистрларга нисбатан аниқланиши мумкин.

4-мавзу. **Программларни кўчириш**

Асосий саволлар:

1. *Программани тезкор хотирага юклаш.*
2. *Бошқарув секциялари ва программаларни боғлаш.*
3. *Ассемблерда адресни узатиш буйруқлари.*
4. *Ассемблерда арифметик амаллар.*

Таянч ибора ва тушунчалар: *операндлар, адреслар майдони, модификаторлар, бошқарув секциялари.*

1. Программани тезкор хотирага юклаш.

Кўп ҳолларда бир вақтда бир нечта программани бажариш имкониятига эга ва бу программалар оператив хотира ва машинанинг бошқа қурилмаларидан ўзаро тақсимланган ҳолда ишлатиш мақсадга мувофиқ бўлади. Бунинг учун программани оператив хотиранинг ихтиёрий жойига юклаш имконияти бўлиши керак ва бу ҳолда программа адреси у хотирага юклангунга қадар номаълум бўлади. Юқорида келтирилган 1-программа абсолют программага мисол бўлади. Бу программа тўғри ишлаши учун у хотиранинг 1000 адресидан юкланиши шарт, акс ҳолда операндлар адреси программа чегарасидан ташқарига чиқиб кетади. *Мисол учун,*

55 101 LDA THREE 00102D

Бу буйруқдаги операнд адреси 102 D. Фараз қилайлик, программа 2000 адресдан юклансин, у ҳолда 102D адресидан кутилган қиймат бўлмайди.

қуйида программани кўчиришга мисол келтирилган.

Адрес	Объект коди	Буйрук операнд	Адрес	Объект коди	Буйрук операнд	Адрес	Объект коди	Буйрук операнд
0000 ... 0006 ... 1036 … 1076	…. …. 4B101036 …. B410	….. ….. JSUB RDREC ….. RDREC	5000 ….. 5006 …. 6036 … 6076	….. ….. 4B106036 ….. B410 …..	…. …. JSUB RDREC ….. RDREC	7420 … 7426 … 8456 … 8496	… …. 4B10845 6 … B410 ……	… … JSUB RDREC …. RDREC

Кўриниб турибдики, бу программа туғри ишлаши учун буйуқларнинг адреслар майдонига ўзгартиришлар қилиш керак. Шу билан биргаликда айрим буйруқлар (бевосита операнд буйруқлар) ўзгармасдан қолиши зарур. Ассемблерга пограммани юкланиши адреси олдиндан номаълум бўлса ҳам, у юклагичга программа хотирага юкланётган пайтда программанинг қайси қисмларига ўзгартириш қилиш зарурлигини кўрсатиши мумкин. Бундай маълуматга эга объект программага кўчувчи программа дейилади. СЎМ/қВ учун тузилган 2-программада START қиймати 0 қилиб олинганлиги ва 30-формат буйруқларини асосан *Pc* регистрига нисбатан аниқланганлиги хотиранинг ихтиёрий жойига юкланишидан қатъий назар бу буйруқлар тўғри бажарилишини кафолатлайди, фақат 4-формат буйруқларининг адрес майдонига ўзгартириш килиш зарур. СЎМ/ҚВ учун тузилган 2-программа мисолида кўчувчи программа объекти программани ҳосил қилишда ассемблер амал қилиши керак бўлган ишларни келтирамиз.

1. JSUB буйруғи учун объект кодни ҳосил қилишда адрес майдонига RDREC нишонининг программа бошига нисбатан силжишини жойлаштиради.
2. Ассемблер юклагич учун JSUB адрес майдонига программа бошланғич адреси қўшилиши кераклиги ҳақида буйруқ тайёрлайди.

Шу мақсадда программа объект кодига янги ёзув – модификатор ёзуви қўшилади.

Устунлар.

1	2-7	8-9
M	Модификация қилунувчи буйруқ адрес майдонинг программа бошига нисбатан адреси	Модификация қилунувчи буйруқ адрес майдонинг ярим байтлардаги узунлиги

Мисол учун, JSUB буйруғи учун модификатор ёзув

M^00007^05

Бу буйруқ программа бошига нисбатан 00007 адресдан бошланувчи ва узунлиги 5 ярим байтли адрес майдонига программа бошланиш адресини (қийматини) қўшиш кераклигин билдиради: 4B101036 буйруғининг 01036 майдонига программа бошланғич адреси қўшилади.

3. Бошқарув секциялари ва программаларни боғлаш.

Бошқарув секцияси (БС) –программа қисми бўлиб, у ассемблер жараёнидан кейин ўз алоҳидалигини сақлаб қолади ва бошқаларга боғлиқмас равишда хотирага юкланиши ва кўчирилиши мумкин. Одатда БС қисм программа бўлади. БС лар бирорта программанинг мантиқий боғланган қисм программалар бўлганлиги сабабли уларни боғлашни амалга ошириш воситалари бўлиши керак. *Мисол учун* БСдаги буйруқлар иккинчи БСдаги буйруқ ёки берилганлар соҳасига мурожаат қила олиш керак. Бундай мурожаатларни оддий усулда амалга ошириб бўлмайди, чунки ассемблер жараёнини амалга ошириш пайтда ҳар бир БСнинг ўртасидаги боғланишлари (кўрсатгичлар) ташқи кўрсатгичлар дейилади. қуйида СЎМ/ҚВ учун кўрилган программани БСларга ажратилган варианти келтирилган.

Сатр номери	Жойлашув адреси	Нишон	Буйруқ	Операнд	Объект коди	Изоҳ
5	0000	COPY	START	0		Файл нусҳасини олиш .
6			EXTDEF	BUFFER,BUFEND, LENGTH		
7			EXTDEF	RDREC,WRREC,		
10	0000	FIRST	STL	RETADR	172027	қайтиш адресини сақлаш
15	0003	CLOOP	JSUB	RDREC	4B100000	Берилганларни ўқиш.
20	0007		LDA	LENGTH	032023	Ўқиш тугаганлигини аниқлаш.

Программа учта БСдан иборат; асосий программа ва ўқиш ва ёзиш қисм программалари. COPY-биринчи БС номи. У 109 қаторидаги CSECT ифодасигача. CSECT янги БС бошланишини билдиради. Юклагич томонидан қайта ишланиши учун ташқи кўрсатгич (EXTREF) ва ташқи номлар (EXTDEF) сифатида эълон қилиниши керак. COPY БСдаги BUFFER, BUREND ва LENGTH нишонлар COPYдан ташқарида хам ишлатилиши учун улар EXTDEF рўйхатида бўлиши керак. COPY, RDREC, WDREC номларнинг EXTDEF ёзувида эълон қилиниш шарт эмас, улар юклаш жараёнида автоматик равишда аниқланади.

Ташқи номлар аниқловчиси (EXTDEF) формати.

Устунлар ва қийматлар			
1 D	2-7 Ташқи ном идентификатори (шу секцияда аниқланган)	8-13 Номнинг нисбий адреси(16 саноқ тизимида)	14-73 8-13 устунлардек

Ташқи кўрсатгич ёзуви (EXTDEF) формати.

Устунлар ва улар қийматлари		
1 D	2-7 Бу секцияда мурожаат бўлган ташқи номлар	8-73 2-7 устунлардек

Энди программанинг ассамблер томонидан қайта ишланишининг ўзига хос томонларини программанинг 15-қатордаги буйруғи учун код ҳосил қилинишида кўрайлик.

15 0003 CLOOP JSUB RDREC 4B10000

Бу ерда RDREC ташқи кўрсатгич. Ассамблер учун RDREC хотирага юкланиш адреси номаълум.Шу сабабли буйруқ кодининг адрес қисмига 0 қиймати ёзилади. Худди

шундай 160 буйруқнинг адрес майдони ва 190 қатордаги –ҳисобланувчи қиймат 0 бўлади. Юклагич бундай буйруқлардаги адресларнинг “бўш” ўринларини тўлдириш учун модификатор ёзувига қўшимчалар киритиш зарур:

Устунлар ва улар қийматлари				
1	2-7	8-9	10	11-16
M	Модификатция қилинувчи адреснинг программа бошига нисбатан силжиши	Модификатция қилинувчи адрес майдонинг ярми байтлардаги узунлиги (16 саноқ тизимида)	Модификатция ıати (қ ёки ғ)	қиймати кўрсатилган майдонга қўшиладиган ёки айриладиган ташқи номлар

Бошқарув секцияларидан ташкил топган программа матнига мос келувчи обьект программа:

```
 D^BUFFER^000033^BUFEND^001033^LENGTH^00002D
R^RDREC^WDREC
T^000000^1D^172027^4B100000^032023^290007^4B100000^...

M^000004^05^RDREC
M^000011^05^WDREC
M^000024^05^ WDREC
E^000000

H^RDREC^000000^00002D
R^BUFFER^LENGTH^BUFEND
T^000000^1D^B410^B400^B440^77201F^E3201B^332FFA^DB2015^A004^33200
M^000018^05^BUFFER
```

4. *Ассемблерда адресни узатиш буйруқлари.*

Бу турдаги буйруқлар ўзгарувчиларнинг адресларини узатиш имкониятини беради.

LEA бажарувчи адресни юклаш буйруғи. Бу буйруқ хотира ячейкасининг силишини ихтиёрий 16 байтли умумий регистр, кўрсатгич регистри ёки индекс регистрларидан бирига узатади. У қуйидаги кўринишга эга:

LEA 16 битли регистр, 16 битли хотира

бу ерда 16 битли хотира WORD туридаги атрибутга эга бўлиши керак. OFFSET амалини ишлатадиган MOV буйруғидан фарқли, 16 битли хотира операнди LEA буйруғида индексли бўлиши мумкин ва натижа ихчам адреслаш имконини беради. *Масалан,* агар DI регистрида 5 турган бўлса, у ҳолда LEA BX, TABLE [DI] буйруғи TABLE+5 силжишни BX регистрига юклайди.

LDS берилганлар сегменти регистри ва кўрсатгичини юклаш буйруғи. Бу буйруғи хотирадан 32 битли иккиланган сўзни ўқиб, унинг биринчи 16 битни берадиган регистрга, қолган 16 битини DS берилганлар сегменти регистрига юклайди. У қуйидаги кўринишга эга:

LDS 16 битли регистр, 32 битли хотира.

бу ерда 16 битли регистр- ихтиёрий 16 битли умумий регистр, 32 битли хотира бҳлса DOUBLEWORD атрибутли хотира ячекаси. Одатда 32 битли хотира DD псевдобуйруғи билан аниқланган бўлади.

Масалан,
```
HERE_FAR  DD  HERE
LDS   BX, HERE_FAR
```
буйруқлари ёрдамида HERE нишонли блок номери адреси ва силжишини BX ва DS регистрларига юкланади. Шундай қилиб юқоридаги кўринишдаги битта LDS буйруғи қуйидаги узатиш буйруқларининг ўрнини босади:

```
MOV  BX,OFFSET  HERE
MOV  AX,SEG   HERE
MOV  DS, AX
```

LES қўшимча сегмент регистри ва кўрсатгични юклаш буйруғи. Бу буйруқ LDS буйруғига ўхшаш, лекин блок номерини ES га юклайди.

5. *Ассемблерда арифметик амаллар*

8088 МП 8 ва 16 разрядли ишорали ва ишорасиз берилганлар устида тўртта асосий арифметик буйруқларни бажаради: *қўшиш, айириш, кўпайтириш ва бўлиш.* Бунда иккилик ишорасиз сонлар диапазони 0 дан 255 гача (8 битлилар учун) ва 0 дан 65538 гача (16 битлилар учун) ўзгаради. Ишорали иккилик сонлар диапазони -128 дан 127 гача (8 битлилар учун) ва -32768 дан 32767 гача (16 битлилар учун) ўзгаради. 8088 МП ўнли сонларни жойлаштирилган (тахланган) ва жойлаштирилмаган (тахланмаган) шаклда сақлайди. Бунда ҳар бир байт иккилик- ўнлик кодлардаги 2 рақамини ўзида сақлайди. Бунда жойлаштирилган ўнли соннинг ҳар бир байти иккилик- ўнлик коддаги иккита рақамдан иборат бўлади ва 00 дан 99 гача қийматни қабул қилади. Жойлаштирилмаган ўнли сон ҳар бир байти фақат битта иккилик-ўнлик коддаги рақамни ўз ичига олади ва 0 дан 9 гача қийматни қабул қилади.

4.1. қўшиш буйруқлари.

ADD қўшиш буйруқлари. Кўрниши: ADD қабул қилувчи, узатувчи.
Одатда бунда қабул қилувчи ва узатувчи сифатида регистр ва хотира ячейкаси қатнашиши мумкин. Бунда буйруқ бажарилгандан сўнг, қабул қилувчи қ қабул қилувчи қ узатувчи ҳосил бўлади. Шу билан бирга узатувчи сифатида буйруқдан тўғридан-тўғри келтирилган берилганлар ҳам келиши мумкин.

Масалан:

```
ADD AL, DL
ADD AL,  3
ADD AL, CATS
ADD DX, [SI]
ADD AL, CATS [SI]
ADD AL,  CATS[BX+SI]
ADD  AL, CATS [BX+SI+10]
ADD  CATS, AL
ADD  CATS[SI], AL
ADD CATS[BX+SI], AL
ADD CATS[BX+SI+(5*80)], AL
```

4.2. ADS кўчиришли қўшиш буйруғи.

Бу буйруқ худди ADD га ўхшаб иш бажаради, лекин қўшиш пайтида кўчириш байроғи CF ни ҳам ишлатади, яъни қўшишдан сўнг қабул қилувчи қ қабул қилувчи қ узатувчиқ кўчириш ҳосил бўлади. Бунда агарда қўшилувчи қийматлар натижаси 8 битга сиғмаса, сиғмаган *бит* CF байроғига ўрнатилади.

Масалан, 250 ва 10 сонларини қўшайлик.

```
   1111 1010 (250 сонининг иккилик кўриниши)
+
   0000 1010 ( 10 сонининг иккилик кўриниши).
   _________
1    0000  0100 ( жавоб: ўнли 260 қиймат).
```

Лекин ишорасиз бутун сонлар диапазони 0 дан 255 гача эди. Шунинг учун биринчи 8 битда турган қиймат қабул қилувчи регистрда жойлашади, 9 битдаги қиймат эса CF байроғига ўрнатилади. Бундан кўриниб турибдики, 8088 МП икки хил қўшиш буйруғини ишлатади. Уладан бири ADD байтларни ёки сўзларда берилган қийматларни ҳамда юқори

аниқликда қийматларнинг кичик қисмларини қўшиш учун хизмат қилади. ADS бўлса, юқори аниқликдаги қийматларнинг катта қисмлаини қўшиши учун хизмат қилади.

Масалан, агарда операндларнинг қиймати 16 битдан ошадиган бўлса, у ҳолда қуйидаги буйруқлар кетма-кетлигидан фойдаланилади.

```
ADD  AX, CX        ; аввал кичик 16 битлар қўшилади.
ADC BX, DX         ; сўнгра катта 16 битлар қўшилади.
```

Улар бажарилганда CX ва DX ларда жойлашган 32 битли сон AX ва BX ларда жойлашган 32 битли сон билан қўшилади. Бунда ADC буйруғи (CX)+(AX) қўшишдан ҳосил бқлган ихтиёрий қўчиришни (DX)+(BX) га қўшади.

4.3. Айириш буйруқлари

8088 МПда айириш икки хил ишорали сонларни қўшиши каби амалга оширилади. Бунинг учун айирилувчининг ишораси ўзгартирилади. *Масалан,* 10-7 қуйидагича ёзилади: 10+(-7) .

SUB айириш буруғи. Бу буйруқ ишлатилганда қуйидагича амал бажарилади:

қабул қилувчиққабул қилувчи-узатувчи

Мисол: SUB AX,CX - SBB қўчиришни ҳисобга олиб айириш буйруғи. Бу буйруғ ҳам SUB буйруғига ўхшаб ишлатилади, лекин қўшимча CF кўчиш байроғининг қийматини ҳам айиради, яъни

қабул қилувчиқ қабул қилувчи-узатувчи – қўчириш.

Мисол: SBB BX, DX

Агарда операндлар узунлиги 16 битдан ошиб кетса, у ҳолда қуйидаги кетма-кет буйруқларни ишлатиш мақсадга мувофиқ бўлади:

```
SUB  AX, CX        ; кичик 16 битлар айирилди.
SUB BX, DX         ; сўнгра катта 16 битлар айирилди.
```

Бу ерда CX ва DX ларга жойлашган 32 битли AX ва BX жойлашган 32 битли сондан айирилади. SBB буйруғи айириш пайтида “ қарз” олиб туришни ҳисоьга олади. Айири буйруқларини қуйидаги хиллари ишлатилади:

```
SUB AX, MEM_WORD
SUB MEM_WORD [BX], AX
SUB AL, 10
SUB MEM_BYTE, OFN
```

SUB ва SBB буйруқлари ишлатилганда худди қўшиш буйруқларида бўлгани каби байроқларга (CF, PF, AF, ZF, SF, OF) аломат ўрнатилади. Худди қўшишдаги каби айиришда ҳам 8088 МПда операндларни иккилик сонлар деб қарайди. Шунинг учун ҳам BCD (иккилик-ўнлик коддаги) сонларни айришда натижа нотўғри чиқиб қолиши мумкин. Буни олдини олиш учун натижани тузатиш буйруқлари AAS ва DAS лардан фойдаланиш керак бўлади. AAS буйруғи ASCII кодида берилганларни айришдаги натижани тузатиш бўлса, DAS ўнли формада берилганларни айришдаги натижани тузатиш учун хизмат қилади. Иккала буйруқда ҳам тузатиладиган сон AL регистрида турибди деб ҳисобланади.

Назорат саволлари

1. Кўчувчи программа хотирага юклаиншда қандай авзаллик бериши мумкин?
2. Программанинг кўчувчи эканлигининг билдирувчи аломат нима ?
3. Кўчувчи программада буйруғи тўғри бажарилиши учун қайси формат буйруқлари ўзгартирилиши керак?
4. Нима сабабдан бошқарув секциясида нисбий адрес 0 дан бошланади ?
5. Секциясида 4 формат буйруғи адрес майдонига 0 ёзилишига сабаб нима ?
6. LEA бажарувчи адресни юклаш буйруғининг вазифаси нимадан иборат?
7. ADD қўшиш буйруқи ва унинг вазифалари.

8. ADS кўчиришли қўшиш буйруғини тушинтиринг.
10.Айириш буйруқлари ва уларга мисоллар келтиринг.

5-мавзу. **Макропроцессорлар**

Асосий саволлар:

1. *Ассемблерда арифметик амаллар.*
2. *Макроаниқловлар ва макрокенгаймалар.*
3. *Ассемблер учун MACRO ва MEND дериктивалари.*
4. *Макропроцессор.*

Таянч ибора ва тушунчалар: *макропроцессор, макроаниқлов, макрокенгаймалар.*

1. Ассемблерда арифметик амаллар

1.1. Кўпайтириш буйруғлари.

MUL ва IMUL кўпайтириш буйруқлари. MUL буйруғи ишорасиз сонларни, IMUL бўлса бутун ишорали сонларни кўпайтиради.Буйруқлар байтларни ва сонларни кўпайтириши мумкин.

Кўриниши: MUL узатувчи
IMUL узатувчи

Бу ерда узатувчи-хотиранинг байт ёки сўз ўлчамидаги ячейкаси. Иккинчи операнд сифатида AL (агар амал байтларда бўлса) ёки AX (агар амал сўзларда бўлса) регисторлари ишлатилади.Байтларни кўпайтирганда 16 битли кўпайтма AH (катта байтлар) ва AL (кичик байтлар) регисторларида ҳосил бўлади.

Сўзларни кўпайтирганда 32 битли кўпайтма DX (катта сўзлар) AX (кичик сўзлар) регисторларида ҳосил бўлади. MUL буйруғи бажарилганда CF ва OF байроқлари 0 бўлади, агар кўпайтманинг катта қисми 0 бўлса, акс ҳолда CFқ1 ва OFқ1.

IMUL буйруғи бажарилганда CFқ0 ва OFқ0 бўлади, агар кўпайтманинг катта қисми кичик қисмининг ишорасини кенгайганини ташкил этса, акс ҳолда CFқ1 ва OFқ1.

AAM-ASCII кодида бериганларни кўпайтириш натижасини тузатиш буйруғи.Бу буйруғ кўпайтувчи ва кўпайтирилувчиларда турган тўғри жойланмаган байтларни кўпайтиришдан чиққан натижани иккита тўғри жойланмаган ўнли операнда кўринишига келтирилади. Бунда кўпайтма AH ва AL регистерларида туради деб ҳисобланади. Буйруқ бажарилганда AL нинг қиймати 10 га бўлиниб, бутун ва қолдиқ мос равишда AH ва AL да сақланади. Бундан ташқари AL даги натижага кўра PF , ZF ва SF байроқларнинг қиймати ўзгаради.

Мисол: AL да 9 (0000 1001B), BL да 7 (0000 0111B), турган бўлсин.

MUL BL бажарилганда AL нинг қиймати BL нинг қийматига кўпайтирилади ва 16 битли натижа AH ва AL регисторларида ҳосил бўлади. Бизнинг ҳолимизда AH да 0, AL да 0011 1111 B (ўнли 63) ҳосил бўлади. Ундан кейинги AAM буйруғи AL нинг қийматини 10 га бўлиб, бутунни 0000 0110B AH регистрига қайтаради, қолдиқ 0000 0011 Вни AL регистрига қайтаради. Демак биз жойланмаган 63 BCD сонни ҳосил қиламиз.

8088 МП да ўнли жойланган сонларни кўпайтириш йўқ, бунинг учун аввал сонлар жойланмаган кўринишга келтирилади, сўнгра кўпайтирилади, ҳосил бўлган натижа қайтадан жойланади.

1.2. Бўлиш буйруқлари.

DIV иккита ишорасиз сонни, IDIV ишорали бутун сонларни бўлишда ишлатилади.Уларнинг кўриниши : DIV узатувчи IDIV узатувчи. Бу ерда узатувчи – бўлувчи ёки байт ёки сўз бўлиб, хотиранининг бир ячейкасида ёки ишчи умумий регистридан бирида бўлади. Бўлинувчи иккиланган ўлчамли бўлиши керак., чунки у AH ва AL (8 битли сонларни бўлиш), DX ва AX (16 битли сонларни бўлиш) регистрларидан олинади. Аввалда узатувчи операнд байт бўлса, қолдиқ AH да, бутун қисми AL регистрида ҳосил бўлади. Агарда узатувчи операнд сўз бўлса, қолдиқ DX да, бутун қисми

AX да ҳосил бўлади. Иккала буйруқ ҳам байроқларга таъсир кўрсатмайди, лекин AL ёки AX га сиғмай қолса, 8088 МП 0 га бўлиш узилишни ишлаб чиқади.

Мисол:
```
DIV  DX
DIV  MEM_BYTE
IDIV  DL
IDIV  MEM_WORD
```

ADD ASCII кодида берилганларни бўлганда тузатиш буйруғи. Бу буйруғ олдинги ўнли тузатиш буйруқлари (AAA, DAA, AAS, DAS, AAM) амал натижасига кўра иш бажаради. Буларга тескари ADD буйруғи *бўлиш амали* бажарилишидан олдин бажарилади. ADD буйруғи жойланмаган бўлувчини иккилик қийматига ўзгартириб, уни AL га жойлайди. Бунинг учун бҳлинувчининг (АН дагининг) катта рақамини 10 га кўпайтиради ва ҳосил бўлган натижани AL да турган кичик рақамга қўшади.

Макроаниқловлар (макрокўрсатмалар) бошланғич программалаш тилидаги кўп ишлатиладиган ифодаларнинг ёзишнинг ўнғай воситасидир. Макропроцессор тамонидан макрокўрсатмаларни мос ифодалар гуруҳига алмаштирш жараёнини макрокенгайтма ёки *макрогенерация* деб аталади. Макрокўрсатмалар программа тузувчиларга ўз программасини ихчам кўринишда ёзишга имкон беради, программанинг якуний матнини ҳосил қилишни макропроцессор ўз зиммасига олади. Умуман олганда, макропроцессорларнинг асосий вазифаси белгиларнинг бир гуруҳини иккинчисига алмаштиришдир. Айрим махсус ҳолатларни ҳисобга олмаганда макропроцессор ўзи қайта ишлаётган матн мазмунини таҳлил қилмайди. Макропроцессор қурилиши ва имкониятига фойдаланилаётган программлаш тилидаги ифодаларнинг шакли таъсир қилиши мумкин, лекин бу ифодалар мазмуни уларни машина кодига трансляцияси, макрогенерация жараёнига бевосита таъсири йўқ. Шунинг учун макропроцессор ишлаш механизми амалда ўзи керак бўлган машина тузилишига боғлиқ эмас.

Одатда макропроцессорлар ассемблер тилида программалашда нисбатан кўп ишлатилади. Бироқ юқори босқич программалаш тилларида, операцион тизим буйруқ тилларида, тадбиқий программа тизимларида ҳам макропроцессорлардан фойдаланиш катта самара беради. Бундан ташқари бирор бир аниқ тил билан боғланмаган мақсадли макропроцессорлар мавжуд.

Макроаниқловлар ва макрокенгайтмалар. СЎМ/ҚВда учун ёзилган программани макрокўрсатмалардан фойдаланилган варианти қуйидаги расмда келтирилган.

Сатр номери				Изоҳ
5	COPY	START	0	Файилдан нусха кўчириш.
10	RDBUFF	MACRO	&INDEV,BUFADR	
15		CLEAR	X	Такрорлаш санагичини тозалаш.
20		CLEAR	A	
25		CLEAR	S	
30		LDT’	#4096	Ёзувнинг максимал узунлигини ўрнатиш.
35		TD	қX’INDEV’	қурима тайёрлигини текшириш .
40		JEQ	*-3	Тайёр бўлишни кутувчи такрорланиш.
45		RD	қX’INDEV’	Белгини А регистрга ўқиш.
50		CJMPR	A,S	Ёзув тугаганлигини
55		JEQ	қ11	Тугаган бўлса, чикиш.
60		STCH	&BUFEDR,X	Белгини буферга ёзиш
65		TIXR	T	Ёзув максимал узунлигигача такрорлаш
70		JLT	*-19	
75		STX	&RECLTN	Ёзув узунлигини сақлаш
80		MEND		
85	WRBUFF	MACRO	&OUTDEV,&BUFED	
90		CLEAR	X	Такрорлаш санагичини тозалаш

95		LDT	&RECLTN	
100		LDCH	&BUFEDR,X	Буфердан белгини танлаш
105		TD	&OUTDEV,	Курилма тайёрлигини текшириш.
110		JEQ	*-3	Тайёр булишни кутувчи таукрорланиш.

Бу программада иккита *макрокўрсатма* —RDBUFF ва WRBUFF аниқланган ва фойдаланилган. Бу макроаниқловлар вазифаси ва мантиқи RDREC ва WRREC қисмпрограммалар мазмуни билан бир хил. Бу ерда ассемблер учун иккита MACRO ва MEND янги директивалар ишлатилган.

MACRO — макроаниқлов бошланишни англатади, сатрдаги нишон майдонидаги эса макроаниқлов номи берилади. Операнд майдонида макроаниқлов рамзий пареметрлари берилади. Барча рамзий номлар "&" белги билан бошланади, бу ҳол *макрогенерация* жараёнида рамзий параметрларни фактик (ҳақиқий) параметрлар билан алмаштиршга имкон беради. MACRO директивасидан кейин макроаниқлов танасини ташкил қилувчи ифодалар келади. Айнан мана шу ифодалар макрогенерация жараёнида ҳосил қилинади. MEND директиваси макроаниқлов тугашини англатади. Асосий программа 135 сатрдан бошланади. 140 сатрдан ифода макроинициализацияни амалга оширувчи макрочақирув буйруғидир. У макропроцессор ҳосил қилиш керак бўлган макрокўрсатма номи, унинг фактик параметрларини аниқлаб беради.

Бу программа макропроцессор тамонидан қайта ишланиши натижаси қуйида келтирилган.

				Изоҳ
5	COPY	START	0	Файлдан нусҳа кўчириш.
10	FIRST	STL	RETADR	қайтиш адресини сақлаш
15	CLOOP	RDBUFF	F1,BUFFER,LENGTH	Буферга ёзувни ўқиш
20		CLEAR	X	Такрорлаш санагичини тозалаш
25		CLEAR	A	
30		CLEAR	S	
35		LDT	#4096	Ёзувўмакгл узунлиги ўрнатиш
40		TD	қ 'X'F1'	қурилма тайёрлигини текшириш
45		JEQ	* — 3	Тайёр бўлиш кутувчи такрорниш
50		RD	+ 'X'F1'	Белгини А регистрга ўқиш
55		COMPR	A,S	Ёзув тугаганлигини текшириш
60		JEQ	* S11	Тугаган бўлса, чиқиш
65		STCH	BUFFER,X	Белгинибуферга ёзиш
70		TIXR	T	Ёзув максимал узунлигигача
75			*— 19	такрорлаш.

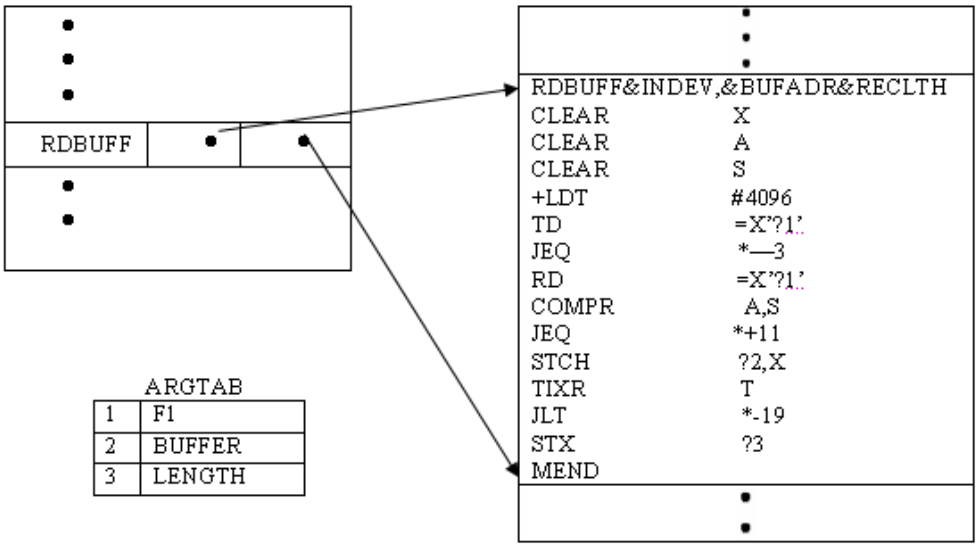
Натижавий программа матнида макрокўрсатмалар йўқ, чунки уларга ҳожат қолмайди. Ҳар бир макрочақирув макроаниқлов танаси билан алмашган ва унда рамзий параметрлар чақирувдаги фактик параметрлар билан алмаштирилган. Эътибор берилса, макроаниқловда нишонлар ишлатилмаган. Макропроцессор тамонидан қайта ишланган программа (файли) ассемблерига кириш файли сифатида объект программани ҳосил қилиш учун берилиши мумкин.

Макропроцессор. Жадваллар ва мантиқ. Икки ўтишли макропроцессорли тасаввур қилиш қийин эмас: биринчи ўтишда макроаниқловлар қайта ишланади, иккинчи ўтишда барча макроинициадлизациялар бажарилади. Бироқ икки ўтишли макропроцессор бир макроаниқлов иккинчи макроаниқлов ичида бўлишига йўл қўймайди (чунки барча мароаниқловловлар инициализациялардан олдин бажарилиши мумкин). Айрим ҳолларда бир макроаниқлов ичида иккинчисини ишлаш самарали (фойдали) бўлиши мумкин. Бир ўтишли макропроцессор учта асосий берилганлар тузилишидан фойдаланилади.

DEFTAB — макроаниқлов жойлашадиган жадвал бўлиб, макроаниқловни ташкил қилувчи унда макро турлар ва ифодалар жойлашди, қайта — ишлаш самарадорлигини

ошириш учун ифодаларда рамзий параметрлар уларнинг тартиб номерлари билан алмаштирилган. NAMTAB жадвалида жойлашади, мазмунига кўра бу жадвал DEFTAB жадвалига кўрсатгичдир. Ҳар бир макроаниқлов NAMTAB жадвалида макроаниқловнинг DEFTAB жадвалига бошланиши ва охирги кўрсатгич бўлади.

Учинчи жадвал бу ARGTAB —– макроинициадлизация ифодаси қайта ишланаётган пайтда тўлдириладиган жадвал. Макроинициадлизациялар аниқлангандан кейин, унинг аргументлари (фактик параметрлари) ARGTAB жадвалига уларнинг жойлашув тартиб номери билан жойлаштиради. Макрогенерация пайтида макроаниқловдаги рамзий параметрлар ARGTAB мос аргументлар билан алмаштирилади.



Ичма-ич жойлашган макроаниқловларни тўғри аниқлаш учун макропроцессор LABEL санагичини ишлатади. Ҳар бир MACRO калит сўзи учраганда LABEL қиймати биттага ошади, учраса MEND биттага камаяди. LABELқ0 бошланғич MACRO га мос MEND учраганлигини англатади. Аксарият макропроцессорлар нисбатан кўп ишлатиладиган макроаниқловларни ўз ичига олган стандарт тизимли кутубхоналарни ишлатишга йўл қўяди. Бу ҳолда программа бошланғич матнида бундай макроаниқловлар ёзишга ҳожат бўлмайди, улар макропроцессор тамонидан зарур пайтда кутубхонадан олинади.

Назорат саволлари .

1. Макро ва қисмпрограмма тушунчаси ўртасидаги фарқ нимада?
2. Макроаниқловда формал параметрлар номида & белгиси нимани англатади?
3. Нима учун макропроцессор бир ўтишли ?
4. Макроаниқловларда ичма-ич жойлашиш даражаси қанча бўлиши мумкин ?

6-мавзу. **Юклагичлар ва боғловчи программалар**

Асосий саволлар:

1. *Боғловчи юклагич алгоритмлари ва жадваллари.*
2. *Юклагичнинг машинага боғлиқмас хусусиятлари.*
3. *Юклаш жараёнини бошқариш.*
4. *Программаларнинг оверлей структукаси.*

Таянч ибора ва тушунчалар: юклагич, трансляторлар, кутубхоналар, оверлей структураси.

Юклагич—бу тизим программаси бўлиб, у объект программани машина хотирасига юклаш вазифасини амалга оширади. Аксарият юклагичлар бундан ташқари кўчириш ва боғланишларни ҳам амалга оширади. Одатда боғлаш функцияси юклаш ва кўчириш вазифаларидан ажратилади. Боғлаш – боғлаш программаси (боғлаш тахрири) орқали, кўчириш ва юклаш—юклагич орқали бажарилади. *Трансляторлар* (ассемблер ва компляторлар) ҳар бир конкрет тизимда бирор стандарт шаклдаги объект кодларни яратадилар. Шу сабабли юклагич ва боғлаш программалари учун программа матни қайси тилда ёзилгани аҳамиятсиз. Бу маърузада юклагичнинг асосий вазифаларидан бири бўлган объект программани оператив хотирага ёзиш ва унинг бажарилувчи биринчи адресига бошқарувни узатишни кўриб чиқамиз. Юклагичлар икки турга бўлинади: *абсалют* ва *боғловчи* юклагичлар. Соддалаштирилган ўқув машинаси учун ёзилган ассемблер программасини хотирага юклаш ва унга бошқарувни бериш учун абсолют юклагич тўғри келади. СЎМ учун ёзилган программага мос объект программа қўйидаги кўринишга эга бўлсин:

```
H^COPY^0010000^00107A
T^001000^1E^141033^482039^001036^281030^301015^482061^3C1003^00102A^0C1039^00102D
T^00101E^15^10C1036^482061^081033^4C0000^454F46^000003^ 000000
T^002039^1E^041036^001030^E0205D^30203F^D8205D^281030^302057^549039^2C205E^38203F
T^002057^1C^101036 ^4C0000^F1^0010000^ 041030^E02079^302064^509039^DC2079^2C1036
T^002073^07 ^382064^382064^4C0000^05
E^001000
```

Абсолют юклагичнинг иши нисбатан содда. Барча иш бир ўтишда амалга оширилади: биринчи навбатда берилган программа тўғрилиги текширилади. *Масалан*, программанинг оператив хотирага сиғиши ёки йўқлиги, бунинг учун бош қисм ёзуви ўқилади. Кейин навбат билан программа танаси ёзувлари ўқилиб, кўрсатилган адрес бўйича жойлаштирилади ва ниҳоят, тугаллаш ёзуви ўқилгач программанинг бошланғич адресига бошқарув узатилади.

Боғловчи юклалич алгоритмлари ва жадваллари. Боғловчи юклалич алгоритми абсалют юклагич алгоритмига нисбатан мураккаб. Чунки унда обьект программалар ўртасида боғланиш ташқи номларда кўрсатгичларни аниқлашга боғлиқдир. Бу кўрсатгичлар ташқи номларга адреслар аниқлангандан кейингина қиймат қабул қилади. Шу сабабли боғловчи юклагич худди ассемблерга ўхшаб иккита ўтишда амалга оширилади. Биринчи ўтишда ташқи мурожаатлар учун адресларни аниқлайди, иккинчи ўтишда юклагични, кўчириш ва боғлашларни бажаради. Боғловчи юклагич учун ташқи номлар жадвали –ESTAB зарурдир. Бу жадвал SYMTABга ўхшаш бўлиб, унда номлар ва ташқи марожаатлар адреслари (барча бошқарув секциялари учун). Бошқа муҳум ўзгарувчилар –PROGADDR (программа юклагич адреси) ва CSADDR (бошқарув секцияси адреси). PROGADDR – оператив хотирадаги боғланувчи программа юкланиши зарур бўлган программа бошланиш адресидир. Юклагич бу адресни операцион тизимдан олади. CSADDR-айни пайтда юклагич томонидан қайта ишланаётган бошқарув секциясининг бошланиш адреси. Бу адрес бошқарув секциядаги борча нисбий адресларга қўшилади. Боғловчи юклагич биринчи ўтишда обьект программанинг фақат бош қисм ёзуви ва аниқловчи ёзувни қайта ишлайди. Биринчи бошқарув секцияси учун PROGADDR бошланғич адрес бўлади. Бошқарув секцияси ёзувларидан ўқилган номлар (секция номи , аниқ ёзувдан номлар) ESTABга қийматлари билан киритилади. Бу адреслар аниқловчи ёзувдан қийматлар CSADDR га қўшишдан хосил бўлади. Тугаллаш ёзувини ўқигандан кейин бошқарув секция узунлиги (CSLTH) CSADDR га қўшилади ва бу қиймат кейинги бошқарув секцияси учун бошланғич адрес хисобланади. Биринчи ўтишдан кейин ESTAB борча ташқи номлар ва улар қийматига эга бўлади. Программани

юклаш, кўчириш ва боғлаш амалда иккинчи ўтишда қилинади. CSADDR ўзгарувчиси айни пайтда оператив хотирага юкланаётган бошқарув секциясининг бошланиш адреси бўлади. Программа танасининг навбатдаги ёзуви ўқилганда, ундан боьект код кўрсатилган адресга юкланади (бу адресга CSADDR қиймати қўшилади). Модификатор ёзув учраганда модификатцияда ишлатилган ном ESTAB дан қидирилади ва унинг қиймати кўрсатилган адресга қўшилади ёки айрилади. Юклагич ўз ишини юкланган программага бошқарувни бериш билан тугаллайди.

Юклагичнинг машинага боғлиқмас хусусиятлари

Бу бўлимда юклагичнинг машинага боғлиқмас томонларини кутубхоналарнинг программаларини излаш (стандарт программани фойдаланиш), ташқи номларни ўзгартириш ва йўқотиш, ташқи номларни автоматик қайта ишлаш ва программаларни оверлей юклаш масалаларини кўрамиз.

Кутубхоналарда автоматик излаш. Аксарият боғловчи юклагичлар юкланаётган программа кутубхоналардан қисмпрограммаларни автоматик равишда қўшиш имкониятини беради. Бу холда программа тузувчидан фақат бундай программаларни номларини ташқи номлар рўйхатига кўрсатиш талаб қилинади. Бу механизмга *кутубхонали излаш* деб аталади. Кутубхонали излашни амалга ошириш учун юклагичлар маьлумот оқимида аниқланмаган ташқи номлар рўйхатига эга бўлиши керак. Бунинг учун аниқловчи ёзувдаги номлар уларнинг адреси аниқланмасдан олдин ESTAB жадвалига киргизилади ва махсус белги билан белгиланади. Ном қиймати (адреси) учраганда, бу қиймат ESTABга киритилади ва шу ном тўлиқ аниқланган хисобланади.

Биринчи ўтишдан кейин ESTABдан қийматлари аниқланмаган номлар ҳал қилинмаган ташқи мурожаатлар ҳосил бўлади. Уларни ҳал қилиш учун юклагич берилган барча кутубхоналарни қараб чиқиш керак (кутубхоналар маьлумот оқимининг бир қисми сифатида қаралади). *Мисол* тариқасида SQRT функциясига муролжаатни олишимиз мумкин. Кутубхоналарни ташкил қилишда махсус файл тузулмасидан фойдаланилади. Бу тузилиш каталог (директорияга) эга бўлиб, унда борча файллар номлари ва уларнинг диск файлида жойлашган адреслари бўлади. Айрим операцион тизимлар доимий кутубхоналар каталогига эга бўлиб, бу ҳол излаш жараёнини тезлаштиради.

Юклаш жараёнини бошқариш. Юклагичлар юклаш жараёнини ўзгартириш имконини берувчи бошқарув параметрини ишлатиши мумкин.

Бошқарув параметрларини бериш учун маҳсус бошқарув тилидан фойдаланилади.

Буруқлар .

INCLUDE <программа номи > (<кутубхона номи >)- бу буйруқ юклагичга кўрсатилган программани кутубхонадан ўқиш ва уни кирувчи оқимда маьлумот сифатида ишлатишни билдиради. Навбатдаги буйруқлар БС ёки ташқи номларнинг рўйхатдан чиқариш ва ташқи номларни ўзгартириш имкониятини беради:

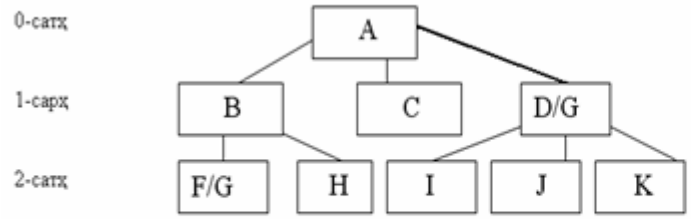
DELETE <секция — номи> — маьлумотлар оқимида кўрсатилган секцияни қараб чиқмаслик (ҳисобдан чиқиш) буйруғи CHANGE <1-ном>, <2-ном>-<1 — ном>ни <2-ном>га алмаштириш. *Мисол учун*, COPY, WRREC, RDREC БС бўлган программани кўрайлик. Бу учта БС бита файлда (кутубхонада) жойлашган бўлсин. Фараз қилайлик , программа тузувчи COPY программада RDREC ва WRREC БС ўрнига UTLIB хизматчи программалар тўпламидан READ ва WRITE қисм программаларини ишлатишмоқчи. Бунинг учун программа матнини ўзгартириш қилиш ёки юклагичга қуйидаги маҳсус буйруқлар кетма-кетлиги берилиши мумкин:

```
INCLUDE READ (UTLIB)
INCLUDE RWITE (UTLIB)
DELETE RDREC, WRREC,
CHANGE RDREC, READ
CHANGE WRREC, RWITE
```

LIBRARY <кутубхона номи>—қўшимча кутубхоналарни оқимга киритиш буйруғи. Бу кутубхоналар стандарт кутубхоналардан олдин қаралади.

NOCALL<ташқи номлар рўйхати>- юклагичга рўйхатдан ташқи номлар ҳал қилинмай қолиши керак эканлигини билдиради. Бу буйруқ оператив хотирани тежайди ва юклагич ишини енгиллаштиради.

Программаларнинг оверлей структукаси. қуйидаги расмда 11 бошқарув секциясидан ташкил торган оверлей структурали программасининг схемаси (дарахти) ва секчичлар узунликлари келтирилган. Бу ерда Ағилдиз БС, B,C ёки DFE секцияларини чақириши, B секция FFG ёки H чақириши мумкин. DFE ёки FFG биргаликда ишлайдиган секцияларни белгиланиши. Дарахт тугунларига сегментлар дейилади. Илдиз сегмент программа иш бошлашидан олдин юкланади ва программа тугагунча хотирада қолади. Бошқа сегментлар уларга мурожаат бўлгандагина хотирага юкланади. Бир вақтда оператив хотирага юкланган сегментлар актив сегментлар дейилади. Мисол учун H БС божараётган бўлсин , H ни B чақиради. B ни A , демак A, B, H сегментлари актив дейилади.



Бошқарув секцияси	Узунлиги.
A	1000
B	1800
C	4000
D	2800
E	800
F	1000
G	400
H	800
I	1000
J	800
K	2000

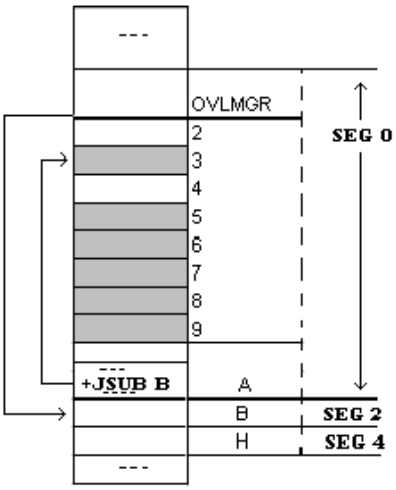
Бир сатҳдаги сегментлар бир вақтда бажарилмайди, шу сабабли улар хотиранинг айни бир сохасидан фойдаланиши мумкин. Агар бошқарув бирорта сигментга берилса, у ўзи билан бир сатҳдаги сегментнинг оператив хотирадаги ўрнини эгаллайди, натижада программа бажарилиши учун ишлатиладиган оператив хотира программа умумий хажмидан кам бўлади. *Оверлей* принципидан фойдаланишнинг туб мохияти ҳам шундадур. Оверлей структураси программа юклагич буёруғи ёрдамида берилади.

сегмент	Бошланғич адрес		Узунлиги
	Нисбий	Амалда	
1(A)	0000	8000	1000
2 (B)	1000	9000	1800
3(C)	2800	A800	1400
4(D,E)	2800	A800	800
5(F,G)	1000	9000	4000
6(H,G)	1000	9000	3000
7(I)	4000	C000	1000
8(J)	4000	C000	800
9(K)	4000	C000	2000

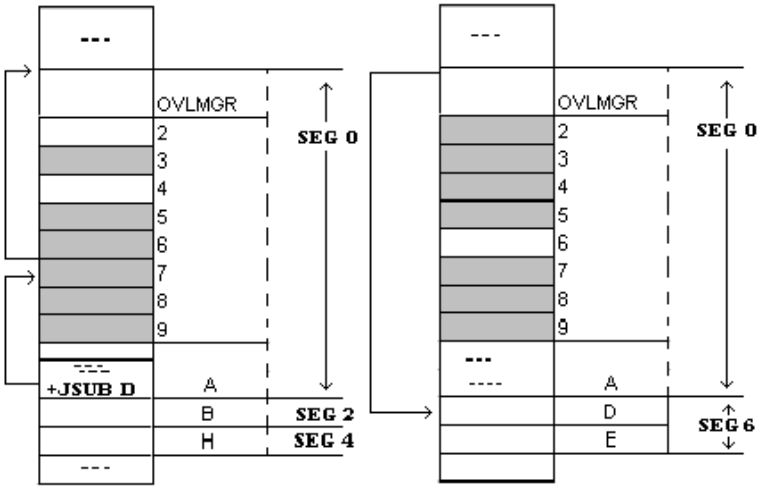
```
SEGMENT SEG 1(A)
SEGMENT SEG 2(B)
SEGMENT SEG 3(F,G)
PARENT SEG 2
SEGMENT SEG 4(H)
PARENT SEG 1
SEGMENT SEG 5(C)
PARENT SEG 1
SEGMENT SEG 6(D,E)
    SEGMENT    SEG   7(I)
        PARENT SEG 6
    SEGMENT    SEG 8 (J)
        PARENT SEG 6
```

SEGMENT <сегмент номи> (< бошқарув секцияси >). Бу буйруқ сегмент номини ва унга кирувчи БС рўйхатини аниқлайди. Буйруқлар кетма –кет жойлашуви аҳамиятлидир. Биринчи аниқланадиган сегмент, кейинги сегментлар "ота-бола" муносбатини аниқлайди. PARENT буйруҳига кўрсатилган сегментлар учун *ота* сегмент эканлигини билдиради. Юқорида оверлей тузулишини (дарахтни) аниқловчи буйруқлар кетма-кетлиги келтирилган. Оверлей программалар учун хотира тақсимоти улардаги ота-бола шажаравий ҳолатига қараб тақсимланади. Оверлей программаларнинг асосий хусусияти уларда хотирага юкланмаган программага мурожаат қилишдадур.

Программа ишлаш жараёнида сегментларни хотирага юклашни махсус программа оверлей менеджери амалга оширади. Бу программа алоҳида бошқарув секциясида (мисол учун OVLMGR) жойлашади. OVLMGR программа структураси тўғрисидаги маьлумотларга эга бўлиши керак. Бу маьлумотлар сегментлар жадвали SEGTAB бўлади. SEGTABни юклагич яратади ва ўзак каталогининг алохида бошқарув секцияси сифатида қўшиб қўяди. SEGTAB программа сегментлар шажарасини ва ҳар бир сегмент учун улар кириш нуқтасини ва SEGFILE даги жойлашуви ҳақида маьлумот беради.



Актив сегментга (SEG 2) мурожаат



Актив булмаган сегментга (SEG 6) мурожаат

Бундан ташқари SEGTABда ўзак сегментдан (А) ташқари сегментлар учун бошқарувни узатиш сохаси бўлади. Бу сохада керакли сегментга бошқарувни узатиш буйруқларини ўз ичига олади.

Назорат саволлари

1. Абсолют юклагич программа хотиранинг қайси адресидан юкланиши кераклигини қандай аниқлайди ?
2. СЎМ учун тузилган программани боғловчи юклагич билан хотирага юклаш мумкинми ?
3. Нима сабабдан боғловчи юклагич бир ўтишда программани хотирага юклай олмайди ?
4. Юклагичлар қандай қилиб охирги вазифаси – программанинг биринчи буйруғига бошқарувни узатишни амалга оширади ?
5. Ҳал қилинмаган ташқи номлар нимани англатади?
6. Паскал тилидаги стандарт кутубхоналардан изланадиган номларга мисол келтиринг .
7. Программа бажарилишида қайси холда оверлей менеджери ишга тушади?
8. Бир сатҳдаги сенментлар учун оператив хотирада ажратиладиган соҳа ўлчами қандай аниқланади?

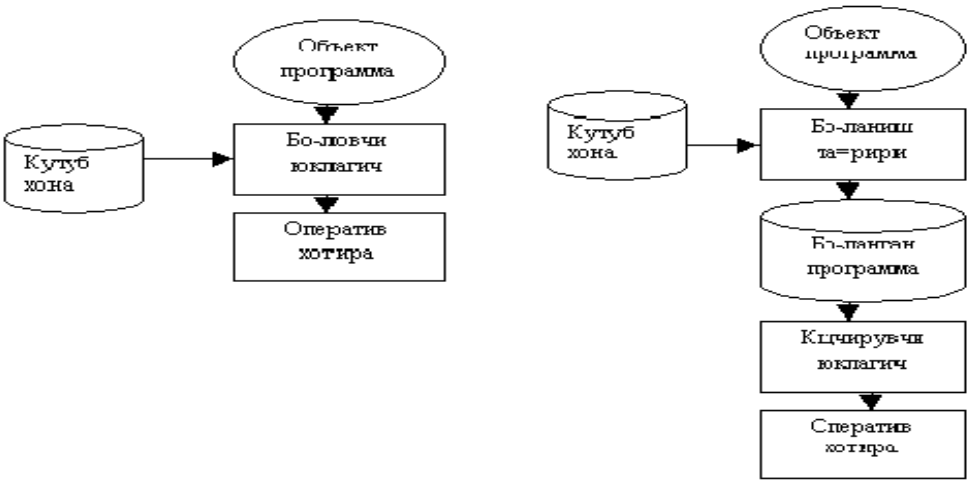
7-мавзу. **Боғланишлар таҳрири**

Асосий саволлар:

1. *Программанинг бажарилувчи модуллари.*
2. *Динамик боғланишлар.*
3. *Компиляторлар.*
4. *Бекус-Науре шаклидаги грамматика.*

Таянч ибора ва тушунчалар: *боғловчи юклагич, бажарилувчи модуллар, компилятор, лексима, тил семантикаси.*

Боғлавчи юклагич боғланишларни ўрнатиш, кўчириш керак бўлса кутубхонадаги автоматик излашларни амалга ошириб, тайёр программани бевосита оператив хотирага юклайди. Боғланишлар таҳрири боғловчи юклагичдан фарқли равишда ташқи боғланишлари ҳал қилинган программа вариантларини тайёрлаб юкланувчи модул ёки бажарилувчи программаларни яратади ва уларни файл кўринишда дискга ёки кутубхоналарга ёзиб қўяди.



Бундай программани юклаш учун оддий кўчирувчи юклагичдан фойдаланиш мумкин. Юклагичга барча нисбий адресларга программа бошланғич адресини қўшиши етарли, яъни модификация қилиниши керак бўлган барча адреслар программа бошланғич адресига нисбатан қиймат олади ва ташқи номлар жадвалига ҳожат қолмайди. Юқоридаги расмда боғловчи юклагич ва боғланишлар таҳрири ўртасида фарқ кўрсатилган. Боғланишларни ўрнатишнинг шундай усули борки, унда қисмпрограммага боғланиш шу программага биринчи мурожаат вақтида амалга оширилади. Бундай боғланишга *динамик боғланиш* ёки *динамик юклаш* дейилади. Юклагич ҳақида гапирганда «Юклагичнинг ўзи оператив хотирага қандай юкланади?» деган савол туғилиши мумкин. Агар юклагич операцион тизим томонидан юкланади десак, операцион тизимни нима ёки ким юклайди? Бу саволга жавоб вариантлари қуйидагича: агар машина хотираси «бўш» деб ҳисобласак, у ҳолда илк программанинг юкланиш адреси сифатида бирорта ўзгармас абсолют адресни олишимиз мумкин. Одатда бу программа сифатида ОС бўлиб, у олдиндан белгиланган жойни эгаллайди. Демак, бизга бу ишни амалга ошириш учун абсолют юклагич керак. Юклагичнинг объект кодини оператив хотирага оператор тамонидан «қўлда» киритилиши мумкин, бошқа усулда абсолют юклагич оператив хотирада доимий равишда резидент сифатида сақланади (доимий хотира қурилмаси ДХҚ).

Маълум бир сигналлар келганда «тизим сатри» ишга тушади. Бунда ДХҚ дан абсолют юклагич оператив хотирага ўқилади ва ишга тушади ёки ДХқни ўзида

бажарилади. Абсолют юклагич вазифаси ташқи қурилмадан битта ёзувни ўқиб бошқарувни унга беришдир. Бу ёзувда юклаш жараёнини амалга оширувчи буйруқлар бўлади. Агар бу буйруқлар бита ёзувга сиғмаса ҳар бир ёзувда юклаш жараёнини амалга оширувчи буйруқлар бўлади ва ҳакоза. Биринчи юкланувчи ёзувни одатда ўраб олувчи юклагич дейилади. Бундай юклагичлар бўш машинада ишлаш мўлжалланган ҳар бир объект программа бошланишида бўлиши керак. Одатда бундай программалар сифатида операцион системанинг бошланғич юклагич программалари бўлади.

Компиляторлар

Компиляторлар ишланишини тушуниш учун Паскал тилида тузилган программани кўрамиз. Одатда компиляторлар қуришни осонлаштириш учун юқори босқич тиллари қандайдир грамматика атамалари орқали тавсифланади. Мълумки, ҳар қандай тил грамматикаси тилда рухсат этилган жумлалар шаклини (синтаксисини) аниқлайди.

Компиляциянинг асосий вазифаси программа тузувчи тамонидан бирор грамматикада ёзилган жумлаларни тил тузилмаларига мослигини аниқлаш ва ҳар бир жумлага мос код генерация қилиш. Қуйидаги программани кўрайлик:

```
1. Program Stats;
2. Var Sum,SumSQ,I,Value,Nean,Variance:integer;
3. Begin
4. Sum :қ0;
5. SumSQ:қ0;
6.  For I:=1to 100 do
7.  Begin
8.  Read (Value);
9. Sum:=Sum +Value;
10. SumSQ:= SumSQ + Value * Value
11. End;
12. Mean:қSum  div 100
13. Variance:= SumSQ DIV 100 – Mean* Mean;
14. Write (Mean, Variance);
15. End.
```

Программанинг бошланғич матнида жумлаларни лексемалар кетма-кетлиги сифатида қараш ўнғайдир. Лексемаларни тилни барпо қиладиган «фундамент –ғишт»га қиёслаш мумкин. *Лексема* –у калит сўз ўзгарувчининг тўлиқ номи, арифметик оператор ва ҳакоза. Лексемаларни аниқлаш ва синфларга ажратиш жараёнига *лексик таҳлил* дейилади. (компиляторларнинг бу қисмига *сканер* дейилади). Лексемалар бир-биридан ажратилгандан кейин программанинг хар бир жумласи тилнинг бирорта қурилмаси деб қабул қилади. *Масалан*, эълон қилиш оператори, қиймат бериш оператори ва ҳакозалар. Компитляторнинг бу вазифани бажарувчи қисмига *синтаксис таҳлил* дейилади. Компиляторнинг энг охирги иши, аниқланган тузилмага мос объект кодни генерация қилишдир.

Грамматика. Программалаш тили грамматикаси программа айрим жумлалари ёки бутун бир программа ёзиладиган тил синтаксиси ёки шаклларининг рамзий тавсифидир. Шуни қайд қилиш керакки, грамматик жумлалар семантикасини (мазмунини) аниқламайди. Семантика ҳақида маълумот объект кодни генерация қилувчи программалардан бўлади. Синтаксис ва семантика ўртасида фарқни кўришга мисол кўрайлик. **I: = J+K ва I:= X+4**

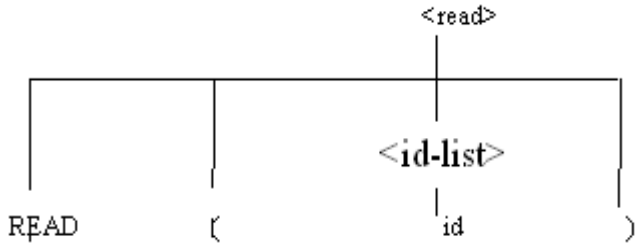
Бунда X, 4 ҳақиқий ва бутун турдаги қиймат ва ўзгарувичлар. Бу иккита жумла бир хил синтаксисга эга— иккаласи ҳам қиймат бериш амалидир, лекин улариниг семантикаси хар хил. Биринчиси бутун сонли арифметикани англатади, иккинчиси сузувчи нуқталарда сонларни қўшиб, натижани бутун сон кўринишига ўтказиш кераклигини англатади. Кўриниб турибдики, бу иккита жумла учун хар хил машина буйруқлари кетма-кетлигига

компиляция қилинади. Демак, бу жумлалар ўртасидаги фарқ объект кодни генерация қилишда кўринади. Грамматикаларни беришнинг хар хил шакллари бор, шуларнинг бари *Бекус — Наура* шакли (формаси) (БНШ). БНШ грамматикаси ҳосил қилувчи қоидалар тўпламидан иборат бўлиб, бу қоидалар қандайдир программалаш тили қурилмасининг синтаксисини аниқлайди.

Паскал тилининг содда грамматикаси.

1. <prog> ::=PROGRAM < prog – name > VAR <dec – list > BEGIN <Stmt – list > END.
2. < prog – name > ::=ID
3. <dec – list > :: <dec >| <dec – list >; <dec >
4. <dec > ::= <id – list > : <type>
5. <type> ::=INTEGER
6. <id – list >::=id | <id–list>, id
7. <Stmt – list >:: =<Stmt >| <Stmt – list >;<Stmt >
8. <Stmt >::=<assign>| <read>|<write>|<for>
9. <assign>|::=id := <exp>
10. <exp>::=<term> |<exp> + <term>| <exp> - <term>
11. <term>::=<factor>| <term> +<factor>| <term> div <factor>
12. <factor>::=id| int|(<exp>)
13. <read>::=Read(<id-list>)
14. <write>::= WRITE(<id –list>)
15. <for> ::=FOR<index- exp> DO <body>
16. <index- exp>::=id ::=<exp> TO<exp>
17. <body>::=<Stmt>| BEGIN <Stmt – list >END.

Мисол учун, <read>::=Read (<id-list>) Паскал тилида Read жумласининг синтаксисини аниқлайди, у грамматикада <read> сифатида , «::=» белгиси «аниқланиш бўйича» деб ўқилади, < > ичидаги белгиларга нотерминал белгилар дейилади (грамматика ичида аниқланадиган қурилмалар номлари). Бурчакли қавсга олинмаган белгилар *терминаллар* дейилади. Юқоридан келтирилган грамматикада <read>,<id-list> нотерминаллар Read ,(,) – терминал белгилар ҳисобланади. Жумлаларин синтаксис таҳлил натижасини дарахт кўринишида ифодалаш ўнғайдир. Бу дарахтларни одатда, грамматик таҳлил дарахтлари дейилади. қуйидаги расмда READ (VALUE) жумласининг грамматик таҳлил дарахти келтирилган.



Паскал тили грамматиканинг 9- қоидаси қиймат бериш амалининг синтаксисини аниқлайди: <assign>|::= id := <exp> . Програмадаги Variance: SumSQ DIV 100 -Mean *Mean жумласига мос келувчи грамматик таҳлил дарахтининг кўриниши қуйида келтирилган. Грамматик таҳлил дарахти ёрдамида хар хил амаллар қатнашган ифодаларни таҳлил қилишимиз осон бқлади. Мисол учун a*b – c Fd ифодани ҳисоблаш амалларини кетма-кетлигини тўғри бажарилишини таъминлаш мумкин.

Назорат саволлари

1. Программанинг бажарилучи модуллари (EXE ва COM туридаги файллар) қандай ҳосил қилинади.?
2. Ҳар доим ҳам боғланишлар таҳрири боғловчи юклагичга нисбатан самаралими?
3. Динамик боғланиш қайси ҳолларда маъқул ҳисобланади?
4. Ўраб олувчи юклагич амали ШЭҲМ да қандай бажарилади?
5. Компилояторнинг 3 та асосий босқичларини санаб беринг .
6. Тил грамматикаси орқали нима берилади: тил синтаксиси ёки тил семантикаси ?
7. Бекус—Науре шаклида берилган грамматикада терминал ва нотерминалларни кўрсатинг.
8. Ифодаларда ҳисоблаш амалларини тўғри бажарилиши кетма-кетлиги грамматика орқали берилиши мумкинми ?
9. Табиий тиллар грамматика орқали тўлиқ берилиши мумкинми ?

8-мавзу. **Лексик ва синтаксис таҳлил усуллари**

Асосий саволлар:

1. *Лексик таҳлил тушунчаси.*
2. *Синтаксис таҳлил тушунчаси.*
3. *Юқоридан-пастга грамматик усули.*
4. *Код генерацияси.*

Таянч ибора ва тушунчалар: лексик таҳлил, рекурсия, генерация, идентификатор.

Лексик таҳлил. Лексик таҳлил — бу компиляция қилунувчи программани «кўздан кўчириш» ва программа матни жумлаларида қатнашувчи лексемаларни таҳлил қилишдир. Компилятор одатда программа бошланғич матнига кирувчи калит сўзлар, операторлар ва идентификаторларни аниқлай олиши керак. Сканер лексемаларни аниқлашда икки хил усулда фойдаланиши мумкин. Идентификатор ва бутун сон каби объектлар яхлит қаралади. Иккинчи усулда ҳар бир лексема грамматика қоидаси билан аниқланади.

<iden>::=<letter>| <iden> <letter>| <iden> <digit>]
<letter>::=A| B|……
<digit>::=0|1|2…..

Сканер ишлаш натижасида лексемалар кетмағкетлиги ҳосил бўлади. Сканер ишлаши тезлигини ошириши учун ҳар бир лексемага сон коди берилади. Программа матнини таҳлил қилиш натижасида жадвал қурилади.

Лексема	Код
PROGRAM	1
VAR	2
BEGIN	3
END	4
END	5
INTEGER	6
FOR	7
READ	8
WRITE	9
TO	10
DO	11
.	12
,	13
:	14
: қ	15
қ	16
r	17
*	18
DIV	19
(	20
)	21
Id	22
Ind	23

Сатр	Лексем тури	Лексем ҳусусияти	Сатр	Лексем тури	Лексем ҳусусияти
1	1		7	7	
	22	^STATS		22	^I
2	2			15	
3	22	^SUM		23	#1
	14			10	
	22	^I		23	#100
	14			11	
	22	^VALUE	8	3	
	14		9	8	
	22	^MEAN		20	
	14			22	^VALUE
	22	^VARIANCE		21	
	13			12	
	6		10	22	^SUM
4	3			15	
5	22	^SUM		22	^SUM
	15			16	
	23	#0		22	^VALUE
	12			12	
6	22	^SUMSQ		22	^SUMSQ
	15		11	15	
	23	#0		22	^SUMSQ
	12			16	

Синтаксис таҳлил. Синтаксис таҳлил бошланғич программа матнидан жумлаларни кўрилаётган грамматика қурилмалари сифатида аниқлашдир. Бу жараённи грамматик таҳлил дарахтини қуриш деб қарашимиз мумкин. Грамматик таҳлил усуллари икки хилга бўлинади: “пастдан-юқорига” ва “юқоридан-пастга” усуллари. Грамматик таҳлилнинг “пастдан-юқорига” усулларидан бири **“***олдин келувчи оператор***”**усули деб номланади. Бу таҳлилда кетма-кет келувчи операторлар жуфтлигидан қайси бири олдин бажариш кераклигини аниқланади. *Мисол учун*, A+B*C-D ифодасини таҳлил қиладиган бўлсак, маълумки, қдан олдин * бажарилади. Демак, қ амали * дан кичик, буни қуйидагича ёзамиз: +>*. Кейинги амаллар жуфтлиги * ва - бўлиб, улар учун * > - муносабат ўринли. Кўрилган ифода учун “олдинги оператор” муносабати қуйидагича: A+B*C-D Демак, B*C амали қолган амаллардан олдин бажарилиши керак. Грамматика дарахти қуриш нуқтаи назаридан * амали +ва - амаллардан пастда жойлашади. Дарахт қуришда, ифода чапдан ўнгга, қўшни операторларга нисбатан юқори даражага эга жумла ости топилгунча давом этилади ва бу жумла ости фойдаланаётган грамматика қоидалари билан англанилади. Шундан кейин жумла таҳлили давом эттирилади. Таҳлил жараёни дарахт илдизи қурилганда тўхтатилади. Грамматик таҳлил учун тил операторларини ўртасидаги <муносабатини аниқлаш керак. Бунда оператор сифатида лексема олинади. Шу жумладан, BEGIN, READ ва id лексемалар ўртасида олдин келиш муносабатини, яъни бу лексемалар программада қандай кетма-кетликда келиши кераклигини аниқлаш зарур. Юқорида келтирилган грамматика учун «олдин келувчи оператор» муносабатлар жадвалининг бир қисми қуйида келтирилган .

	VAR	BEGIN	END	END	INTEGER	FOR	READ	WRITE	TO	DO	;	..	,	:x	id	ind
PROGRAM	≅														<	
VAR		≅									≅	≅	≅		≅	
BEGIN			≅	≅		<	<	<			<				<	
END			>	>							>					
INTEGER	>										>					
FOR															<	
TO										>					<	<
DO	<	>	>				<	<	<		>				<	

Бу муносабатлар қуйидагича бўлиши мумкин: PROGRAM ≅ Var,Begin<For. Лексемалар ўртаисидаги «≅» муносабат иккита лексема бир даражада эканлигини ва улар грамматикада аниқланган битта қурилма таркибига киришини англатади. Лекин лексемалар ўртасида олдин келиш муносабати қаътий эмас, бир вақтнинг ўзида <END ва END>; муносабатлари ўринли бўлиши мумкин. Айрим лексемалар жуфтлиги учун «олдин келувчи оператор» муносабати йўқ. Бу ҳол грамматик тўғри ёзилган жумлаларда бу лексемалар ёнма-ён туриши мумкин эмаслигини билдиради.

Юқоридан –пастга грамматик усули (рекурсив тушиш усули). Бу усулда хар бир нотерминал белги учун алоҳида процедура аниқланади. Ҳар бир процедура ичида бошқа нотерминал белгиларни аниқловчи процедураларга (ўзига ҳам) мурожаат бўлиши мумкин. Процедура кирувчи оқимдан мос нотерминал бошланувчи *сатростини* қидиради, агар бундай *сатрости* топилса, прцедура ишини маваффақиятли тугатганлиги ҳақида хабар беради ва кўрсатгичли аниқланган сатростидан кейинги лексемага қўйилади. Агар процедура сатростини талаб қилинган нотерминал сифатида аниқлай олмаса, у ўз ишини маваффақиятсиз тугатади ва хато ҳақида хабар беради.

Грамматик таҳлилнинг рекурсив процедураларига мисол.

```
Prosedure  Read
Begin
FOUND :=False
If TOKEN=8{READ} then
Begin
Навбатдаги лексемлар ўтилсин
If TOKEN=20{( } then
Begin
Навбатдаги лексемлар ўтилсин
If IDLIST маваффақиятли тугади then
If  TOKEN=21{ ) } then
Begin
FOUND :=True
Навбатдаги лексемлар ўтилсин
```

```
Prosedure IDLIST
 Begin
FOUND :=False
If  TOKEN=22{ID} then
Begin
FOUND :=True
Навбатдаги лексемлар ўтилсин
While(TOKEN=14{, })end
(FOUND =TRVV)do
Begin
Навбатдаги лексемлар ўтилсин
If  TOKEN=22 then
Навбатдаги лексемлар ўтилсин
Else FOUND :=False
End.
```

Мисол учун**,** <read>::=Read (<id-list>) қоидасига мос процедура кетма-кет жойлашгаган Read ва ‘(‘лексемаларни қидиради. Агар излаш маваффақиятли бўлса <id-list> қоидасига мос процедура ишга тушади. Агар бу процедура ҳам ўз ишини маваффақиятли тугатса‘)‘ белгиси қидирилади, агар ‘)‘ топилса, кўрсатгич кирувчи старостдаги ‘)‘ кейинги лексемага кўрсатади.

Код генерацияси

Программа синтаксик таҳлил қилингандан кейин компиляциянинг охирги босқичида программанинг объект коди генерация қилинади. Объект коди генерациянинг усуллари кўп ва турли мураккабликдадир. Бу маърузада нисбатан содда усулни кўрамиз. Бунда *код генерацияси* ҳар бир грамматика қоидасига мос ифодани объект кодини ҳосил қилувчи қисмпрограмма тўплами орқали бажарилади. Бу программаларга *семантик* программалар дейилади, чунки бу программалар тилнинг мос қурилмаларига бериладиган мазмунга мос ишларни бажаради. Танланган сода схемага кўра бу программалар объект кодларини генерация қилади. Бизнинг ҳолда Паскал тилидаги жумла СЎМ учун аниқланган ассемблер тилига ўткзилади. Код генерацияси пайтида маълумотларни сақлаш учун рўйхат ишлатилади. LISTCOUNT ўзгарувчиси элементар ҳисоблагичи учун ишлатилади. S орқлаи объект ҳақидаги маълумот олинади:

S(id) – id –идентификатор хусусияти.

S(id) – id –ном ёки унинг жадвалдан номери.

S(int) – сон – #100

rA —қийматни А регистрга юкланганини билдиради.

қуйида <read>::=Read (<id-list>) қоидасиги мос процедуралар кетма-кетлиги берилган.

```
генерация қилинсин  [KJSUB XREAD].
XREAD учун ташқи кўрсаткичлар ҳосил          {XREAD—ташқи кутубхонада қилинсин
қилинсин аниқланган                            стандарт  ўқиш  қисм программаси}
генерация қилинсин [WORD  LISTCOUNT]
Рўйхатдаги ҳар бир элементи учун
   Begin
   S(item) ни рўйхатдан ўчириш
  [WORD  S(item)] ҳосил қилинсин
end
LISTCOUNT:=0
Программа матнидаги  READ (VALUE)    жумласига мос келувчи
 код:
+JSUB              XREAD
WORD                   1
WORD              VALUE
```

Худи шундай <term>1::= <term>2 *<factor> учун

```
   If  S <term>2= rA  then
   ҳосил қилинсин [MUL S   (<factor> )]
   else  If  S < factor>= rA  then
   ҳосил қилинсин [MUL S   (<term > )]
else
begin
Geta (<term>2)
ҳосил қилинсин [MUL S   (<factor>)]
S (<term>1):=rA
Rega:= <term>1
```

Программа матнидаги Varinse:=SUMSQ DIV 100 —Mean*Mean қиймат бериш операторига мос келувчи объект кодлар:

LDA	SUMSQ
DIV	#100
STA	T1
LDA	MEAN
MUL	MEAN
STA	T2
LDA	T1
SVB	T2
STA	VARIANCE

Назорат саволлари

1. Лексик таҳлилнинг иккита усулларини афзаллик нуқтаи назаридан қиёслаб беринг .
2. Лексик таҳлил босқичида прогтраммадаги қандай хатоларни аниқлаш мумкин?
3. Синтаксик таҳлилнинг рекурсив тушиш усулида «рекурсив» сўзини ишлатишга сабаб нимада ?
4. Синтаксик таҳлилда программалаш тилининг бирорта операторини нотўғри ёзилганини аниқлаш ҳолатларини тушунтириб беринг ?
5. Код генерацияси компьютернинг аппарат тузилишига боғлиқми?
6. қиймат бериш оператори учун берилганлар турига қараб хар ҳил машина кодлари кетма-келиги ҳосил бўлишини тушунтириб беринг .
7. Нима сабабдан юқоридан келтирилган мисоллар учун код генерацияси натижалари ассемблер тили етарли ҳисобланган.

9-мавзу. Операцион тизимларнинг асосий вазифалари ва уларнинг турлари

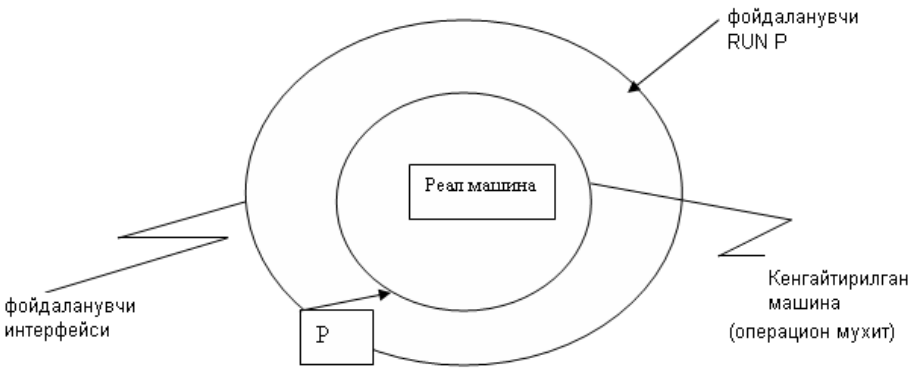
Асосий саволлар:

1. *Операцион тизимларнинг асосий вазифалари.*
2. *Операцион тизимларнинг синфлри.*
3. *ОС нинг машинага боғлиқ хусусиятлари.*
4. *Узилишларни қайта ишлаш.*

Таянч ибора ва тушунчалар: *интерфейси, операцион қамров, мультипроцессор, приоретит.*

Операцион тизимларнинг асосий вазифалари. Операцион тизимларнинг асосий вазифалари – фойдаланувчининг ЭХМ билан мулоқатини соддалаштиришдир. Бундан ташқари, ОС нинг мухим вазифаси- тизимли программа таьминотининг элементлари бўлган *транслятор*, *юклагичлар* ва *мониторлар* учун қулай интерфес яратишдир. Тизимли программа таьминоти (ТПТ) аппарат воситалари устига қурилган ускуртма бўлиб, у фойдаланувчини машина билан ишлашини ўнғай қилади. ЭХМ самарадорлигини ошириш учун ОС ЭҲМ нинг ресурсларини бошқаришдек мураккаб жараёнларни амалга оширади.

ОСнинг асосий вазифаси қуйидаги схемада келтирилган:



ОС билан фойдаланувчи ўртасидан мулоқат маьлум бир бошқарув тилида амалга оширилади (RUN P). ОС нинг фойдаланувчи учун қулайлик нуқтаи-назаридан кўп ишлатиладиган хизматлар учун стандарт сервис программаларига эга. *Масалан*, READ (F) бунда машина даражасида ўқиш-ёзишни амалга оширишни ОС ўз зиммасига олади. Бундай хизматчи программалар ечимдан масалаларнинг операцион қамровининг бир қисми сифатида қаралиши мумкин. Гарчи ОС фақат программа таьминоти кўриниши

амалга оширилган деб хисобланса ҳам, амалларнинг маьлум бирлари микро-программалар тўплами кўринишида программа –аппарат воситалари орқали амалга оширилган .

ОС турлари. ОСлар фойдаланувчига кўрсатилган интерфейс турлари бўйича синфларга ажратилади. Лекин бу ажратиш (қатий) эмас ва синфлар ўзаро кесишиши мумкин. қуйида маьлум бир шартлар (критериялар) бўйича операцион тизимлари синфларга ажратиш келтирилган.

	Критерия	1-синф	2-синф
1	Мулоқат тури бўйича	Пакетли қайта ишлаш тизими	Диологли тизим
2	Фойдаланувчилар сони бўйича	Бир программали тизим	Мультипрограммалик тизим Мультипроцессорлик тизим

Бир программали тизимларда фақат бита фойдаланувчи ишлаши мумкин. Бу ҳол ресурслар чекланганлигидан келиб чиқади. Мультипрограмма тизимида бир вақтда бир неча фойдаланувчи масалалари ечилиши мумкин. Масалалар бир-бирига халақит бермаслиги учун ўзининг операцион қобиғини яратади. Мультипроцессорли тизимларда биттидан ортиқ процессор ишлатилади. Пакетли қайта ишлаш тизимларида топшириқ маьлумот (берилганлар) ташувчиларида ёзилган (дискларда) бошқарув операторлари кетма-кетлиги сифатида келади ва дискларни алмаштириш каби ишлардан ташқари, прграммани ўқиш ва божариш ОС ўз зиммасига олади. Диалог ёки интерактив режимда фойдаланувчиларнинг мурожат вақти тақсимоти тизимлари орқали таьминланади. ОСда фойдаланувчи томонидан берилган ҳар бир буйруқ, у киритилган захоти бажарилади. Ташқи қурилмалардан келаётган сигналларни қайта ишлаш ва тезда жавоб бериш учун *реал вақт* тизимларидан фойдаланилади.

Фойдаланувчи интерфейси. ОС томонидан бериладиган интерфейс турли тоифа одамларга ЭХМ билан мулоқатни ўнғай қилишга қаратилган. Фойдаланувчи учун буйруқлар тили мавжуд бўлиб, унинг энг соддаси менюдир. Мураккаб тизимларда ОС мулоқатнинг бир неча тури бўлиши мумкин. Мутахассис бўлмаган фойдаланувчиларга ЭҲМда ишлаш учун сода буйруқлар тили яратилган. Мутахассис программа тузувчилар учун *масалаларни бошқариш тили* деб номланувчи кучли ва мураккаб тил мавжуд. Булардан ташқари операторлар учун махсус тиллар мавжуд бўлиб, улар бу орқали масалалар бажарилишини тўхтатиш, ҳолатини аниқлаши ва ташқи таьсирларни бошқаришлари мумкин. ОС ва фойдаланувчи интерфейсини қўллаш учун ОС стандарт сервис программаларига эга бўлиши керак. *Масалан*, ШЭХМ да клавиатурадан ўқиш, маълумотларни экранга чиқариш программалари, мураккаб ЭХМларда масофадаги терминал ишини таьминлаш, чоп қилиш қурилмасини бошқариш, локал тармоқ ва ЭХМ лар ўртасидаги боғланишни таьминлаш программалари.

Операцион қамров. ОС энг мухим вазифалардан бири фойдаланувчилар масалаларнинг операцион қамровини таьминлашдир. Бу таьминлаш бир қанча сервис программалардан иборат бўлиб, улар топшириқ (масалани) ечиш жараёнида фойдаланилади ва фойдаланувчи талабига муофиқ машина ресурсларини ажратиш ва бошқаришни таьминлайди. ОС томонидан кўрсатиладиган хизмат сифатида ўқиш-ёзиш фунциясини кўрайлик. СЎМ учун тузилган программада ўқиш-ёзиш учун (бир байтни) такрорловчи жараённи ташкил қилиш керак эди (RD ёки WD). Хатоларни аниқлаш ва бартараф қилиш фойдаланувчи (программа тизувчи) зиммасига юклатилган эди. Ўқиш-ёзишни ОС томонидан бажарилиш масалани кескин осонлаштиради. Бунда стандарт программага керакли қурилма идентификатори ва параметрларини бериш етарли ҳисобланади. Бу турдаги стандарт сервис программалар машинанинг кенгайтмаси

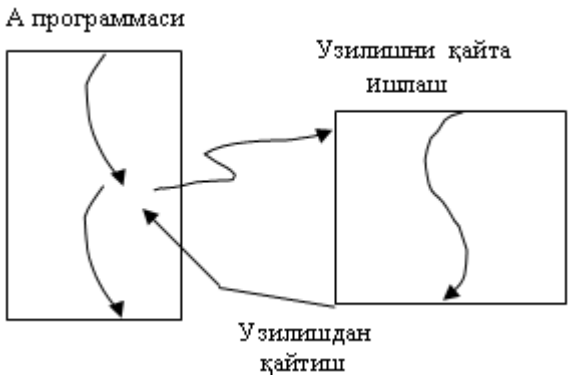
сифатида қаралиши мумкин. Ҳар бир ОС кўп сондаги сервис программаларга эга бўлиб, улар кенгайтирилган машинана ташкил қилади ва фойдаланувчи программаси ишлаш пайтида ишлатилади. Фойдаланувчи программа тузаётганда ОС нинг асосий аппарат кенгайтмаси даражасигача билиш шарт бўлмайди. Айрим ҳолларда кенгайтирилган машина *виртул машина* дейилади, бироқ бу термин бошқа маънода ҳам ишлатилиши мумкин. Мултипрограмма режимли ОС да фойдаланувчи прогрммаси талаби бўйича ЭҲМ ресурсларини тақсимлайдиган стандарт хизматчи программалар мавжуд. Улар фойдаланадиган программа учун оператив хотирани ва марказий процессорни олдиндан аниқланган стратегия бўйича топшириқлар ўртасида тақсимлайди. Айрима тизимларда фойдаланувчилар программалари ОС функцияларини бевосита чиқариши мумкин. Бу ҳолда ОС ҳужжатларида стандарт сервис программалар ва берилганлар соҳалари реал адресларда берилади. *Масалан,* ўқиш-ёзиш программалари хотиранинг 238 адресида жойлашганалигини билган ҳолда фойдаланувчи JSUB 238 буйруғи орқали бу программаларига мурожаат қилиши мумкин. Лекин бу усулнинг хатоликларга олиб келиш эҳтимоли катта ва у ОС ҳимояларини чеклаб ўтади.

Нисбатан такомиллашган ОС да ОС функцияларига мурожаат махсус машина буйруқлари-супервизорни чиқариш (SVC-Super Visor Call) орқали амалга оширилади. Бу буйруқни бажарилиши узилишни чақиради ва натижада бошқарув стандарт сервис программага узатилади. SVC билан бирга бериладиган код ОС сўров турини (функцияни) аниқлайди. Узилиш чақирилганда марказий процессор фойдаланувчи режимидан супервизор режимига ўтади. Бу ҳолатда ОС буйруқларидан ва воситаларидан фойдаланилади. ОС кўп ташкил қилувчилари шу режимда ишлайди. Фойдаланувчи режимида ОС функциялари, марказий процессорнинг ҳимоя байроқлари ўзгартириши ёки бошқа режимига ўтказиши мумкин эмас.

ОСнинг машинага боғлиқ хусусиятлари

ОСнинг асосий (муҳим) вазифаларидан бири ЭҲМ ресурсларини бошқаришдир. Аксарият ресурслар бевосита аппарат қурилмаларига, яъни оператив хотира, ўқиш-ёзиш қурилмалари, марказий процессорга боғлиқдир. Шундан келиб чиқиб айтиш мумкинки, ОС кўп функциялари бевосита ЭҲМ архитектураси билан аниқланади. *Мисол учун,* СЎМда оператив хотирани бошқариш узилишлари, супервизорни чақириш программалари йўқ ва у битта фойдаланувчи билан ишлашга мослашган. СЎМ/қВ да аксинча хотира катта ва унда мультипрограммали ОСдан фойдаланиш мақсадга мувофиқдир ва ресурсларни тақсимлаш имкониятини беради. қуйида биз СЎМ/қВ мисолида ЭҲМ ресурсларини бошқариш муаммосини кўрамиз, айрим ҳолларда маълум бир хоссалар шу турдаги бошқа машиналарга ҳам кўчирилиши мумкин.

Узилишларни қайта-ишлаш. Узилиш (Interrypt)-бу сигнал бўлиб, у ЭҲМни буйруқлар оқимини оддий тартибда бажаришида ўзгариш қилишга мажбур этади. Бу сигналлар ўқиш-ёзиш амаллари бажарилиши ёки олдиндан берилган вақт интервали тугаганда, нолга бўлиш ҳолларида юзага келади. қуйидаги расмда узилишга жавобан бўладиган ҳодисалар кетма-кетлиги кўрсатилган.



Фараз қилайлик, узилиш сигнали келган пайтда А программа бажарилаётган бўлсин. Натижада бошқарув автоматик равишда узилишларни қайта ишлаш блокига берилади, бу блок ОСнинг бир қисмидир. Бу блок узилиш шартига жавоб беришга мўлжалланган. Юқорида қайд қилинган ҳолатда узилишни чақириш А программага умуман боғлиқ бўлмаслиги мумкин. Бу сигнал ўқиш-ёзиш амалини тугаган бошқа программа томонидан чақирилган бўлиши мумкин. Умуман, А программани қайси жойда ва қачон узилишни айтиб бўлмайди, бошқача айтганда, узилиш асинхрон равишда руй беради. Узилган программани кейинчалик тўғри ишлашини аппарат ва тизимли программа воситалари кузатиб боради. Узилиш ҳолатида программа бажарилишига вақтдан бошқа ҳеч нима таъсир қилмайди. қуйидаги расмда СЎМ/қВ учун узилишларнинг тўртта синфи келтирилган.

Синф	Узилиш тури
1	SVC
2	Программавий
3	Таймер бўйича
4	Ўқиш-ёзиш бўйича

1-синф. SVC узилишлари-марказий процессор томонидан супервизорни чақиришда юзага келади. Бу буйруқ программа томонидан ОС функцияларини чақириш учун ишлатилади.

2-синф. Программа узилишли бўлиб у нолга бўлиш, нотўғри машина буйруғини бажаришга уриниш ва бошқа сабаблари бўйича юзага келиши мумкин.

3-синф. Таймер бўйича узилишлар марказий процессорнинг интервал таймери томонидан чақирилади. Узилиш таймер регистрига эга бўлиб, имтиёзли SVC буйруғи орқали қандайдир бошланғич қийматга эга бўлади ва ҳар бир миллисекундан кейин биттага камаяди. Регистр қиймати 0 бўлганда таймер бўйича узилиш рўй беради. Интервал таймеридан ОС фойдаланувчи программаси ЭҲМ бошқарувида қанча вақт қолиш кераклиги аниқлашда ишлатилади.

4-синф. Ўқиш-ёзиш узилишларини ўқиш-ёзиш канали ёки қурилмалари чақиради. Бу узилишларга сабаб ўқиш-ёзиш амалига муражаатдир. Бу узилишлар орқали ўқиш-ёзиш амалини нормал ёки хато билан тугуганлигини билиш мумкин. Узилиш рўй берганида маркзий процессор ҳолати сақланиб қолади ва бошқарувчи стандарт программаларга берилади. Ҳар бир синф узилишлари учун мос узлишлар ишлаш соҳаси ажртилади. *Мисол учун* таймер бўйича узилиш соҳаси 160чи адресдан бошланади. Таймер бўйича узилиш рўй берганда, барча регистрларнинг қийматлари шу соҳада сақланади. Соҳанинг биринчи иккита сўзига олдиндан киритилган қийматларни Sw ҳолат сўзи ва P_c буйруқ ҳисоблагич регистрларига ёзади. Регистрлар қийматини ёзиш ва сақлаш машинанинг аппарат воситалари билан автоматик равишда амалга оширилади. P_c регистрига янги қиймат ёзиш автоматик равишда бошқарувчи мос буйруққа (адреси кўрсатилган) беришни юзага келтиради. Бу адрес таймер бўйича узилишни қайта ишлаш стандарт программасининг бошланиш адресидир. S_w марказий процессорнинг янги ҳолатини аниқлайди. Узилишни қайта-ишлаш стандарт программаси бажарилиши тугаллангандан сўнг, бу программа охирги буйруқ сифатида процессор ҳолатини юклаш буйруғини (LPS-Load Processor Status) ва натижада бошқарув узилган программага берилади. S_w-ҳолат сўзи узилишни қайта ишлаш учун зарур маълумаотларнинг бир қисмини ўз ичига олади. қуйидаги расмда S_w регистрнинг разрядлар бўйича тузилиш келтирилган.

Рзрядлар	Номи	Мазмуни (ишлатилиши)
0	MODE	0-фойдаланувчи, 1-супервизор
1	IDLE	0-актив, 1-пассив
2-5	ID	Процессор идентификатори
6-7	CC	Шарт коди
8-11	MASK	Узилиш маскаси
12-15		Ишлатилмайди
16-23	ICODE	Узилиш коди.

0-разряд. Оддий программа бажарилишида MODEқ0 бўлади. Узилиш рўй берганда MODE=1 бўлади ва марказий процессорни супервизор ҳолатига ўтказади ва имтиёзли буйруқларни бажариш имконияти юзага келади. SWнинг қиймати сақланишидан олдин узилиш сабаби *ICODE*га жойлаштирилади. SVC узилишда *ICODE*да фойдаланувчи SVC буйруғида берган қиймати (операнд) сақланади. *MASK* майдони узилишни ҳал қилиш жараёнини назорати учун ишлатилади. У процессор ҳолатини аниқловчи маълумотларни йўқотиб қўймаслик учун зарур. *MASK* майдонидаги ҳар бир разряд маълум бир синф узилишини аниқлайди. Разрядларни 1 бўлиши мос синф узилишларининг бажарилишига рухсат, 0-ман қилади. Охирги ҳолатда узилиш ниқобланади, яъни *ман* қилинди дейилади. Бироқ бу узилиш йўқолиб кетмайди, у аппарат томонидан эслаб қолинади ва у кечиктирилган деб ҳисобланади. Ҳар бир синф узилишларига *узилиш приоретити* мос қўйилади. Энг юқори приоритетга SVC узилишлари эга, ундан кейин программа узилишлари ва ҳакозо. Шу асосда MASK қиймати ўз приоритетига тенг ва паст бўлган узилишларга ман қилувчи этиб аниқланади. *Масалан*, 1 приоритетли узилишда 1,2 приоритетига рухсат йўқ, ундан юқорига рухсат бор.

Назорат саволлари

1. ОС нинг асосий вазифаси нимада?
2. ОС буйруқ тили ёрдамида қандай ишларни амалга ошириш мумкин?
3. Юқорида келтирилган ОС синфлари жадвали асосида MS DOS ва WINDOWS ОСларни синфларга ажратиб беринг.
4. Программа ишлашида узилиш қайси ҳолатларда рўй беради?
5. Узилиш қайта ишлагандан кейин узилишни чақирган программа иши қандай тикланади ва давом эттирилади?
6. қайси синф узилишлари программа ишлашига боғлиқмас равишда рўй бериши мумкин?

10-мавзу. Жараёнларни режалаштириш

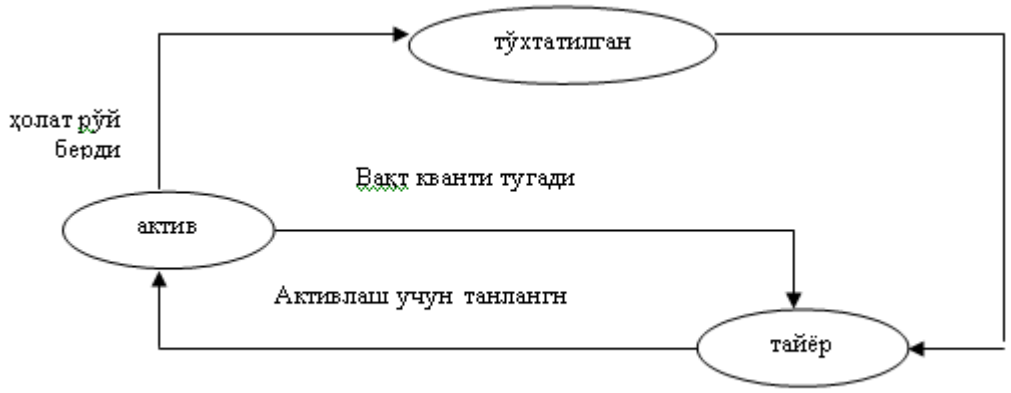
Асосий саволлар:

1. *Жараён ва прграмма ўртасидаги фарқ.*
2. *Жараёнларни амалга ошириш усуллари.*
3. *Ўқиш ва ёзиш жараёнини амалга ошириш.*

Таянч ибора ва тушунчалар: *вақт кванти, машина регистрлари, канал номери.*

Жараён (ёки топшириқ) деб айни пайтда ечимда бўлган программага айтилади. Ҳисоблаш ишларини амалга ошириш учун ОС ҳар бир жараёнга марказий процессорни ажратади. Бир программалик ОСда фақат битта фойдаланувчи жараёни бўлади. Мультипрограммали ОСда марказий процессорга ва бошқа ресурсларга бир нечта ўзаро боғлиқ бўлмаган жараёнлар давогар бўлиши мумкин. Жараёнларни режалаш қандайдир режалаштириш стратегиясига асосан ўзаро рақобат қилувчи бир нечта жараён ўртасида марказий процессор ресурслари тақсимотини бошқаришга айтилади. Жараён фойдаланувчи топшириғи ҳисоблана бошлаганда пайдо бўлади ва топшириқ тугаши билан йўқолади. Жараён мавжуд пайтда 3 ҳолатда бўлади. Жараён *актив* бўлади, агар у

марказий процессордан (ўз буйруқларини бажариш учун) фойдаланаётган бўлса, жараён *тўхтатилган* (блокировка қилинган) дейилади, агар унинг бажарилиши маълум бир шарт бажарилгандан кейингина давом эттирилиши мумкин бўлса. *Тўхтатилмаган* ва *актив бўлмаган* жараёнга *тайёр ҳолдаги жараён* дейилади. Қуйидаги расмда бир ҳолатдан иккинчи ҳолатга ўтиш кўрсатилган:



Жараёнларни бир ҳолатдан иккинчисига қайтиши.

Ҳар қандай вақт моментида битта жараён актив бўлади. Операцион тизим бошқарувини фойдаланувчи жараёнига узатганда интервал таймерини ўрнатади (аниқлайди). Шу орқали топшириқ учун марказий процессордан фойдаланишга ажратилган максимал вақтни- *вақт квантини* аниқлайди. Бу вақт тугаши билан жараён актив ҳолатдан тайёр ҳолатга ўтади. Кейин ОС режалаштириш стратегиясига кўра навбатдаги жараённи танлайди (тайёр ҳолатдаги) ва уни актив ҳолатга ўтказиб, унга бошқарувни узатади. ОСнинг функциясига *диспетчирлик* дейилади. Шундай бўлиши мумкинки, жараён ўзига ажратилган максимал вақтдан тўлиқ фойдаланмаган (ишлатмаган) ҳолда қандайдир шарт бажарилишини кутиш мумкин, бунда жараён тўхташ ҳолатига ўтказилади ва қандайдир бошқа жараён активлашади. Кутилган ҳолат юзага келганда тўхтатилган жараён активлашишига даъвогар бўлиши мумкин. Бир жараён тугагунча бир неча марта *актив-тўхтатилган-тайёр* ҳолатларда бўлиши мумкин. Бу ҳолатлар ҳисоблаш натижасига таъсир қилмаслиги учун унинг ҳолати сақлаб қолиниши керак. ОСда ҳар бир жараённинг ҳолати Жараён ҳолати блокида (PSB) сақланади. Унда жараён қайси ҳолатда бўлганлиги, машина регистрларини сақлаш адреси ва бошқа маълумотлар сақланади (PC ва SW қийматлари).

PSB учун зарур маълумотларни ОТ қуйидагилардан олиши мумкин:

- *Вақт кванти бўйича маълумотни таймер бўйича узилиш соҳасидан;*
- *Бошқарув узатилган жараён ҳақидаги маълумотни SW узилиш соҳасидан олади.*

Диспетчирликда фойдалниладиган алгоритмнинг умумий кўриниши :

Procedure Dispatch;
Актив жараён PSBси янгилансин (актив жараён бўлса);
Навбатдаги тайёр жараён танлансин ва унга бошқарув узатилсин;
If таймер жараён топилган бўлса then
Begin
Танланган жараён Актив деб белгилансин;
STI буйруғи билан вақт кванти ўрнатилсин;
LPS буйруғи билан танланган жараёнга бошқарув узатилсин;
end
else

LPS буйруғи билан марказий процессор бўш ҳолатига келтирилсин.

Бошқарув бир жараёндан иккинчисига узатилганда, шу жараённинг қандай сабаб билан тўхтатилганлигига қараб PSB учун зарур маълумотларни қуйидагилардан олиниши мумкин: вақт кванти тугаган жараёнлар маълумотини таймер бўйича узилиш соҳасидан, бошқарувни берган жараён ҳақида маълумот SVC- узилиш соҳасидан олинади. Навбатдаги жараённи бир нечта усулда амалга ошириш мумкин:

- *Барча жараёнлар бир қийматли (баҳоли) деб ҳисобланади ва циклик танлаш орқали барча PSB қаралади, улар ичидан тайёри танланади. Барча жараёнларга бир хил вақт кванти берилади.*
- *Жараёнлар фойдаланувчи ёки ОТ томонидан берилган приоритетлар (имтиёзлар) асосида танланади.*

Актив жараён кутиш нуқтасига етиб келганда бу ҳақда ОСга хизмат учун сўров – *WAIT* ёрдамида хабар беради (SVC 0). Жараён кутаётган ҳолат учун юзага келганлиги ҳақидаги хабар ОСга *SIGNAL* сўрови (SVC 1) орқали берилади.

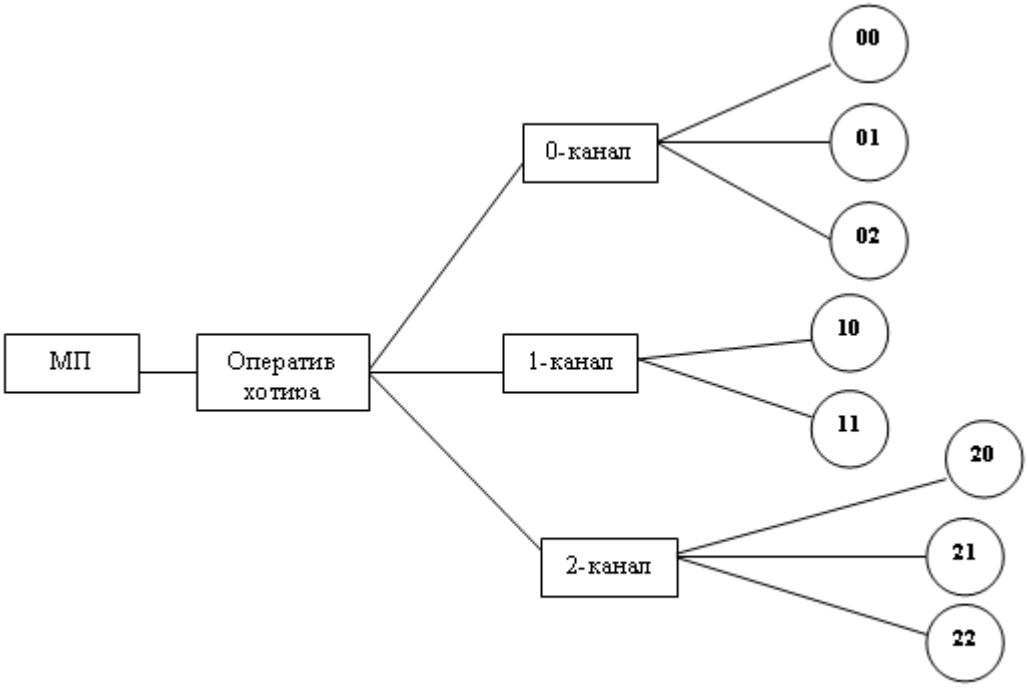
```
Procedure WAIT (ESB)
If  ESBFLAG қ1 then     {ҳолат рўй беради}
LPS  буйруғи ёрдамида бошқарув сўраётган жараёнга қайтарилсин
Else
   Begin
    Сўраётган жараёнини тўхтатилган деб белгилаш
    Жараённи ESBQUE га киритиш
   DISPATCH
   END;
Procedure  SIGNAL (ESB)
ESBFLAG:қ1     {ҳолат рўй беради}
For   ESBQUE   даги ҳар бир жараён  учун DO
    Begin
```

Жараённи тайёр деб белгилаш;

```
        Жараённи  ESBQUE дан ўчириш
    End
        Бошқарувни сўраган жараёнга LPS  ёрдамида қайтариш.
```

Бу ерда ESB (ҳолатни аниқлаш блоки) кутилаётганлиги ёки рўй бериш ҳақида хабар берилган ҳолат блокининг адреси. ESBFLAG- *бит* байроғига эга бўлиб мос ҳолат рўй берган ёки йўқлигини билдиради. Бундан ташқари ESB да ESBQUE – бу ҳолатларни кутаётган жараёнлар рўйхатига кўрсатгич бўлади. WAIT сўрови актив жараён томонидан берилиб, у ESB кўрсатган ҳолат юзага келмагунча жараён давом эта олмаслигини билдиради. Агар ESBFLAG текширилиб, керакли ҳолат юзага келганлиги аниқланса, бошқарув дарҳол жараённинг ўзига қайтарилади. Имтиёзли режалаштиришда WAIT ва SIGNAL бошқача амалга ишлатилади. WAITда мос ҳолатда бошқарувни сўраган жараёнга дарҳол қайтариш ўрнига бошқарувни диспетчерга узатади ва у приоритет бўйича юқори бўлган жараёнларни активлаштиради.

Ўқш ва ёзиш жараёнини амалга ошириш. Мини ЭҲМларда ўқиш-ёзиш байтларда амалга оширилади. *Масалан,* берилганларни ўқиш учун программа ўқиш-ёзиш қурилмаси ҳолатини сўрайдиган ва бир қатор ўқиш буйруқларини бажарувчи циклга эга бўлиши керак. Бунда марказий процессор ўқиш-ёзиш жараёнида бевосита иштирок этади. Такомиллашган ЭҲМларда (мисол учун, ҚВ/СЎМ) ўқиш-ёзиш жараёнининг барча амаллари ва бошқарув ўқиш-ёзиш каналлари орқали бажарилади.



Юқоридаги расмда ўқиш-ёзиш конфигурацияси келтирилган.

Канал бажарилиши керак бўлган амаллар кетма-кетлиги канал программаси орқали берилади. Ўқиш-ёзишни бажариш учун марказий процессор *START I/O* (*SIO*) буйруғини бажаради. Бу буйрукда канал номери ва канал программасининг бошланғич адреси берилади. Кейинчалик канал марказий процессор иштирокисиз ўқиш-ёзиш жараёнини амалга оширади ва бу вақтда марказий процессор бошқа ҳисоблаш ишларини амалга ошириши мумкин. Бир вақтни ўзида бир нечта канал ишлаши мумкин. ОС ўқиш-ёзиш жараёнига бир неча ҳолларда жалб қилиниши мумкин. Тизим фойдаланувчи программаси томонидан ўқиш-ёзишга талабни қабул қилиши ва бу амални қачон бажарилиши ҳақида хабар бериши керак. Иккинчидан, ОС ёзиш каналларини бошқариш ва улар ҳосил қилувчи узилишларни бажариши керак. Фойдаланувчи программаси (ҚВ/СЎМ) да ўқиш-ёзиш амалини бажариш учун SVC 2 буйруғини беради. Буйруқ параметрлари сифатида канал номерини, канал программасининг адреси ва ўқиш-ёзиш тугаганлигини билдиришда ишлатувчи соҳи-*Жараён ҳолати блоки* (ESB) адреси кўрсатилади. Агар программа ўқиш-ёзиш амали тугаллашини кутиш керак бўлса, у SVC 0 (WAIT) буйруғини бажариши керак. Бу буйрукда ҳодиса ҳолати блокининг адреси кўрсатилади. Демак ўқиш-ёзиш амалини бажариш қуйидаги кўринишда бўлади.

SVC 2 {ўқиш-ёзиш амалига сўров}
……..
……..
……..
SVC 0 { натижани кутиш}

Айрим ҳолларда бу иккита буйруқ кетма-кет келиши мумкин. Бироқ, ўқиш-ёзиш жараёни бажарилаётганда программа томонидан бошқа амаллар бажариб туриши (мумкин бўлса) мақсадга мувофиқдир. Қуйида бу жараён тўлиқроқ келтирилган.

```
P1          START 0
… … …
            LDA     #Read   канал программа адреси
            LDS     #1      канал номери
            LDT     #ESB    ҳодиса ҳолати блокининг адреси
            SVC     2       ўқиш тугаллашини кутиш
LOOP        LDA # ESB       ESB адреси
            SVC     0
            ……      { берилганларни ишчи хотирага ёзиш}
            LDA     #0      ESBга  бошланғич қиймат
            STA     ESB
            LDA     #Read
            LDS     #1
            LDT     #ESB
            SVC     2       ўқиш учун навбатдаги сўров
                    ……
                    J       LOOP
```

Бу программада бошқа регистрларга канал программаси адреси, канал номери ва ҳодиса ҳолати блоки адреси жойлаштирилади, кейин SVC юз беради. Канал программаси иккита буйруқдан иборат. 1-буйруқ *ўқиш амали* ва 1 номерли қурилмани аниқлайди ва 256 байт берилганлар BUFIN адресида жойлашиши кераклигини билдиради. Иккинчи буйруқ канал ишини тўхтатиш ва ўқиш-ёзиш бўйича узилишни ҳосил қилади. ESB буйруқ уч байтдан иборат бўлиб, унинг биринчи бити-ҳолат рўй берган (1-ҳа) ёки йўқлигини (0-йўқ) аниқлайди. қолган битларда бу ҳодисани кутаётган жараёнларга кўрсатгич жойлашади. Кутаётган жараёнлар бўлмаса кўрсатгич 0 бўлади. Ўқиш-ёзиш амалига доир мураккаброқ мисол қуйидаги расмда келтирилган. Бу ҳолда 4096 байт ёзувни 22 қурилмадан 14 қурилмага нусха олиш программаси келтирилган. Унда иккита канал программаси бўлиб, уларнинг бири ўқиш учун иккинчиси ёзиш учун ва иккита ҳодиса ҳолати блоки ишлатилади.

```
P2          START0
LOOP        LDA     #0
STA         RDESB
LDA       #READ             22 қурилмага ўқиш учун сўров
LDS         #2
LDT         #RDESB          ўқиш тугашини кутиш
SVC         2
LDA         #WRESB
SVC         0
…….                         Ёзиш кетма-кетлигини ташкил қилиш
LDA         #0
STA         WRESB
LDA         #WRITE          14 қурилмага ёзиш учун сўров
LDS         #1
LDT         #WRESB
SVC         2
J           LOOP
```

Ўқиш учун канал программаси

1-буйруқ

```
READ BYTE   X'12'           READ,
     BYTE   X'100'
     WORD   BUFIN
```

2-буйруқ

```
        BYTE   X'000 000 000 000'
Ёзиш учун канал программаси
1-буйруқ
WRITE  BYTE X'24'
        BYTE        X'1000'
        WORD        BUFOUT
2-буйруқ
        BYTE   X'000 000 000 000'
RDESB   BYTE  X'000 000'
WRESB   BYTE  X'800 000'
BUFIN   RESB   4096
BUFOUT  RESB   4096
```

Бу программада ўқиш-ёзиш амаллари бир-бирига боғлиқмас равишда бажарилади, чунки улар турли каналларни ишлатадилар. Программа ўқиш-ёзиш жараёнларини турли ESBлар алоҳида бошқариб туради. ОС ўқиш-ёзиш амалига сўровларни қандай бажаришни кўрайлик. Бунинг учун ҳар бир ўқиш-ёзиш каналига мос канал ишчи соҳаси ажаратилади. Бу соҳада канал программаси адреси (агар у бўлса), жорий амал учун ESB адреси, ўқиш-ёзиш тугаганлигини билдирувчи байроқ-бит жойлашади. Ишчи соҳа бундан ташқари шу канал учун ўқиш-ёзиш сўровлар кетма-кетлигига кўрсатгич (ОТ стандарт программалари томонидан қўллаб-қувватловчи). Фойдаланувчи ўқиш-ёзиш берилган сўров қуйидагича қайта ишланади. Агар канал банд бўлса, сўров ОС томонидан навбатга қўйилади, акс ҳолда сўров каналга қўйилади. Жорий сўров канал ишчи соҳасида сақланади.

```
Procedure  IOREQ (CHAN, CP, ESB)
Канал   TIO  буйруғи орқали  сўраш
If   канал банд then
(CP, ESB)  канал учун навбатга қўйиш
else
        begin
        (CP, ESB) канал ишчи соҳасида сақлансин
        SIO  орқали канал ишга туширилсин
        End
        Бошқарув сўраган жараёнга LPS  орқали қайтарилсин.
        End
         SVC 2  ни қайта  ишлаш  алгоритми
```

қуйида ОСни ўқиш-ёзиш узилишни қайта ишлаш алгоритми келтирилган.

```
Procudure  IOINTERRUPT (CHFN)
Канал ишчи соҳасидаги байроқни текширсин
IF амал нормал тугаган then
        Begin
        Канал ишчи соҳасидан ESB
        SVS буйруғи билан     ESB орқали ҳодиса рўй берганлигини билдириш
IF канал учун сўров навбати бўш эмас Then
        Begin
                Навбатдан кейинги сўров (CB, ESB) олинсин
                Канал ишчи хотирасида (CB, ESB) сақлансин
                SIO буйруғи билан каналнн ишга туширсин
        End
    End
Else хато тузатиш учун мос амаллар қилинсин
```

Узилган жараёнга LPS буйруғи орқали бошқарув узатилсин.

Агар канал бўш бўлса, у ишга туширилади ва кейин сўров канал ишчи хотирасида сақланади. Бошқарув сўров берган жараёнга қайтарилиши мумкин, ўзаро маълумот алмашиши (ўқиш-ёзиш) бўлаётган пайтда у ўз ишини давом эттириш мумкин. *IOINTERRUPT* процедурасида ўқиш-ёзиш бўйича узилиш бўлганда ОС томонидан бажариладиган ишлар келтирилган.Узилиш чақирган канал номери узилишнинг ҳолат сўзида (узилишнинг ишчи соҳасида) бўлади. Узилиш сабабини аниқлаш учун ОС канал ишчи соҳасидаги байроқ ҳолатини таҳлил қилади. Агар ўқиш-ёзиш нормал тугаган бўлса, узилиш қайта ишловчиси бу ҳодисани ҳолати блоки орқали ОСни хабардор қилади. Бу ички узилишга олиб келувчи *SVC* сўров орқали амалга оширилиши мумкин. Бу ҳолни кутаётган ҳар бир жараён тайёр ҳолатга келтирилади. Агар ўқиш-ёзиш нормал бўлмаган ҳолда тугаса, ОС бу носозликни йўқотиш чораларини кўради: ўқиш хатоси бўлса, қайта ўқишга ҳаракат қилиши; қоғоз тугаган бўлса, операторларга хабар бериш ёки носозлик ҳақидаги хабарни сўраган жараёнга (фойдаланувчи программасига) бериши ва жараён акс таъсирини кутиш мумкин.

Назорат саволлари

1. Жараён ва программа тушунчалари ўртасида фарқ нимада?
2. Жараён тўхтатилган ҳолатдан дарҳол тайёр ҳолатга ўтиши мумкинми? Жавобингизни асослаб беринг.
3. Жараён кейинги активлашувларида ўз ҳолатини қандай тиклайди?
4. Имтиёзли (приоритетли) диспетчерликда тайёр жараёнлар қандай тартибда активлаштирилади?
5. СЎМда битта байтни ўқишни (ёзишни) амалга оширувчи буйруқлар кетма-кетлигини кўрсатинг.
6. Агар битта ўқиш (ёзиш) қурилмасига бир вақт оралиғида бир нечта сўров бўлса, ОС бу муаммони қандай ҳал қилади?
7. қВ/СЎМда ўзгарувчи қиймати тўла ўқилмагунча программа бажарилишини тўхтатиб туриш учун қандай буйруқлар кетма-кетлиги берилиши керак?
8. Турбо-Паскаль тилида ўқиш-ёзиш амаллари қандай бажарилади?

11-мавзу. **Реал ва виртуал хотираларни бошқариш**

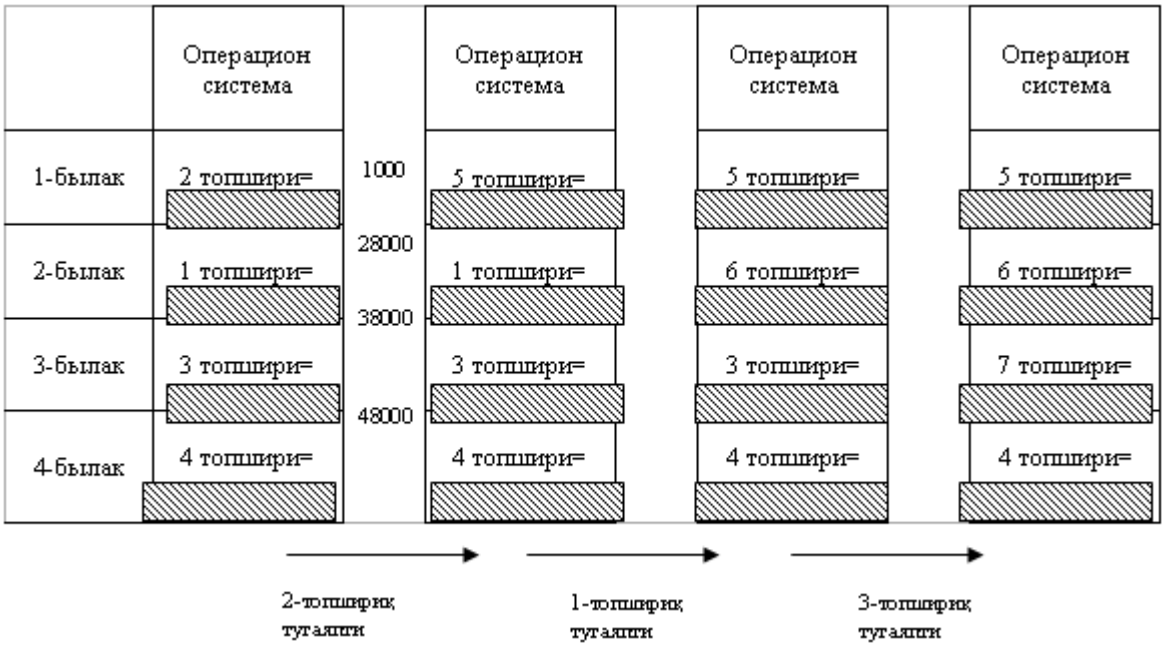
Асосий саволлар:

1. *Реал хотирани тақсимланиши;*
2. *Хотира химоясини амалга ошириш;*
3. *Виртуал хотира ва уни бошқариш;*
4. *Саҳифа узилиши.*

Таянч ибора ва тушунчалар: *реал хотира, виртуал, идентификатор, фиксирланган узунлик.*

Биттдан ортиқ фойдаланувчи бир вақтда қайта ишловчи ҳар бир ОС марказий хотирани биргаликда ишловчи жараёнлар ўртасида тақсимлаш механизимига эга бўлиши керак. Аксарият мультипрограммали тизимларда хотира бўлакларга бўлиниб, ҳар бир жараёнга биттаси мос қўйилади. Бу бўлак ўлчами фиксирланган ёки жараённи бажарилишда динамик равишда аниқланиши мумкин. қуйида хотира тақсимотининг жараёнлар узунликларининг айрим усулларини кўрамиз. Фараз қилайлик 8та жараён ва уларнинг узунликлари берилган бўлсин. ОСнинг мультипрограммалик даражаси, яъни бир вақтда ишлаши мумкин бўлган топшириқлар сони оператив хотирага бир вақтда юкланадиган топшириқлар сони билан чекланади. Марказий хотира ўлчами 50000 байт, ОС 10000 байт жой эгаллайди. Биринчи бўлак бевосита ОСдан кейин бошланадаи ва 18000 байт узунликда 2 ва 3 бўлак 10000 байт ва 4 бўлак 8000 байт жой эгаллайди. Оддий

усул бўйича ОС юкланаётган топшириқни унинг ўлчамига мос келувчи энг кичик ўлчамдаги бўлакка жойлаштиради. Агар бўлак узунлиги топшириқ узунлигидан катта бўлса, бўлакнинг бу қисми ишлатилмайди.



Бизнинг мисолимизда 1-топшириқ 2-бўлакка, 2-топшириқ 1-бўлакка, 3-топшириқ 3-бўлакка, 4-топшириқ 4-бўлакка жойлаштирилади. Ҳар бир топшириқ то тугагунча шу бўлакда қолади. Бу ҳол юқридаги расмда келтирилган. Бўшаган бўлак навбатдаги топшириқ учун ажаратилади ва юкланади. Таъкидлаш зарурки, бўлаклар ўлчами ўзгармасдир. Бўлаклар ўлчамларини қандай қилиб олиш жуда муҳимдир. Катта бўлаклар кўп бўлса, топшириқлпр учун жой етарли, бироқ хотиранинг кўп қисми бўш қолиб кетади. Бу усул топшириқлпр узунликлари конкрет ўлчамлар чегарасида бўлган ҳолларда самарали ҳисобланади. Кейинги расмда бўлаклар ўлчами топшириқлар ўлчамига мос равишда танлаш усули келтирилган, яъни бўлак ўлчами топшириқ ўлчами билан мос келади. Биринчи тақсимлашдан кейин марказий хотира бўлакларга қайта бўлинмайди, 1-топшириқ юкланганда 1 бўлак ўлчами аниқланади ва кетма-кет 2,3,4,5-топшириқлар юкланиб бўлакларга ажратилади. 2-топшириқ тугагандан кейин бу бўлакка 6-топшириқ юкланади. 2-топшириқ ишлатган хотиранинг маълум бир қисми бўш қолиб кетади, у жойга бирорта топшириқ ҳам сиғмайди.

Навбатда турган топшириқ сиғадиган хотира бўлаги пайдо бўлиши билан у хотирага юкланади. Бу нарса бўш хотиралар рўйхати орқали бошқарилади. Хотиранинг қандай тақсимланишидан қатъий назар ОС хотира ҳимоясини таъминлаши керак. Бир бўлакдаги топшириқ бошқа бўлакдаги хотира катакларига ўзгартириш қила олмаслиги керак. Айрим тизимларда бир бўлакдан ташқаридаги хотирадан ўқиш ва фақат ўз бўлагига ёзиши мумкин. Бошқа тизимларда ўқиш-ёзиш фақат ўз бўлагида мумкин қилиб аниқланади.

Хотира ҳимоясини самарали амалга ошириш учун аппарат томонидан қўллаб-қувватлаш зарур. *Мисол учун*, топшириқ бўлагининг бошалниш ва тугашининг адресини сақловчи чегара регистрлари жуфтлигин киритиш мумкин. Бу регистрларга мурожаат фойдаланувчи учун ман қилинади ва уни фақат марказий процессорнинг супервизор ҳолатидагина ишлатиш мумкин. ОС регистрлар қийматларини топшириқ бўлакка юкланганда ўрнатади. Жараёнда ўзгариш рўй берганда бу қийматлар автоматик равишда сақланади. Демак, чегара регистрлари ҳар доим актив жараёнга ажратилган бўлакларнинг адресларини сақлайди. Агар адрес бўлак чегарасидан ташқарида бўлса, хотирага мурожат бўлмайди ва программа узилиш ҳосил қилинади. қВ/СЎМда хотира ҳимоясининг бошқа

усули қўлланилган. Ҳар бир 800 байтлик хотира бўлагига 4 битлик хотира химояси калити мос қўйилади. Фойдаланувчи жараёни 4 битлик жараён идентификаторига эга ва у SW ҳолат сўзининг ID майдонида сақланади. Жараён хотирага юкланганда бўлак блокларининг калитларига фойдаланувчининг жараён идентификаторига ёзиб қўйилади. Жараён хотирага мурожаат қилганда ОТ жараён идентификаторини хотира блокининг калити билан солиштирилади, агар улар мос келмаса мурожаат амалга оширилмайди.

Хотиранинг тақсимлашининг муҳим муаммоларидан бири хотирани самарали бўлинишидир. Жараёнлар ишлашида хотиранинг топшириқ сиғмайдиган ва ўзаро кесишмайдиган бўш хотира бўлаклари пайдо бўлиши мумкин. Бунда бўш бўлаклар йиғиндисига топшириқ сиғадиган тақдирда ҳам у хотирага юкланмайди. Хотирани тақсимлашнинг яна бир усули-бўлакларни силжитишдир. Ҳар бир жараён тугаши билан бошқа бўлаклар хотиранинг бирон чегараси томон сурилади. Натижада бўш бўлаклар чегарада йиғилади ва бу жойга навбатдаги топшириқни юклаш мумкин. Бунда хотира самарали фойдаланилади, лекин бўлакларни кўчириш жуд кўп қўшимча иш талаб қилади ва айрим ҳолларда усул самарасини йўққа чиқариши мумкин. Кўчувчи бўлаклар билан ишлашда программани кўчириш билан боғлиқ муаммолар келиб чиқади.

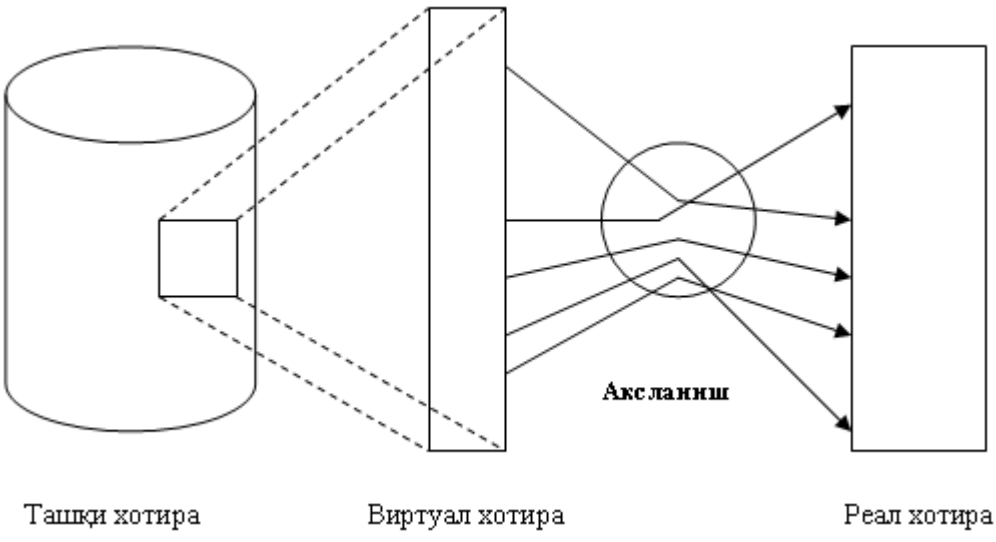
Мисол учун программада 4 формат буйруғи қуйидаги кўринишда бўлсин.

+STA BUFF2 OF108108 (8108 BUFF2)

Бу буйруқ абсолют адрес билан ишлаганлиги учун бўлак кўчгандан кейин бу адресга бошқа бўлакдаги хотира мос келиши мумкин. Бу муаммони ечиш учун кўчиш регистрлари ишлатилади. Бу регистрда бўлак бошланиш адреси сақланади ва буйруқ адреси ҳисобланаётган пайтда қўшилади. Бўлак кўчганда регистр қиймати ҳам ўзгаради.

ВИРТУАЛ хотирани бошқариш

Фойдаланувчи программасига бериладиган ва амалда мавжуд бўлган ресурслардан фарқли хусусиятга эга бўлган ресурсларга **виртуал** дейилади. *Мисол учун*, программага катта виртуал хотирани ишлатишга рухсат бўлсин. Бу хотира ҳажми ЭҲМ ишлатиш мумкин бўлган реал хотирасидан ҳам катта бўлиши мумкин.

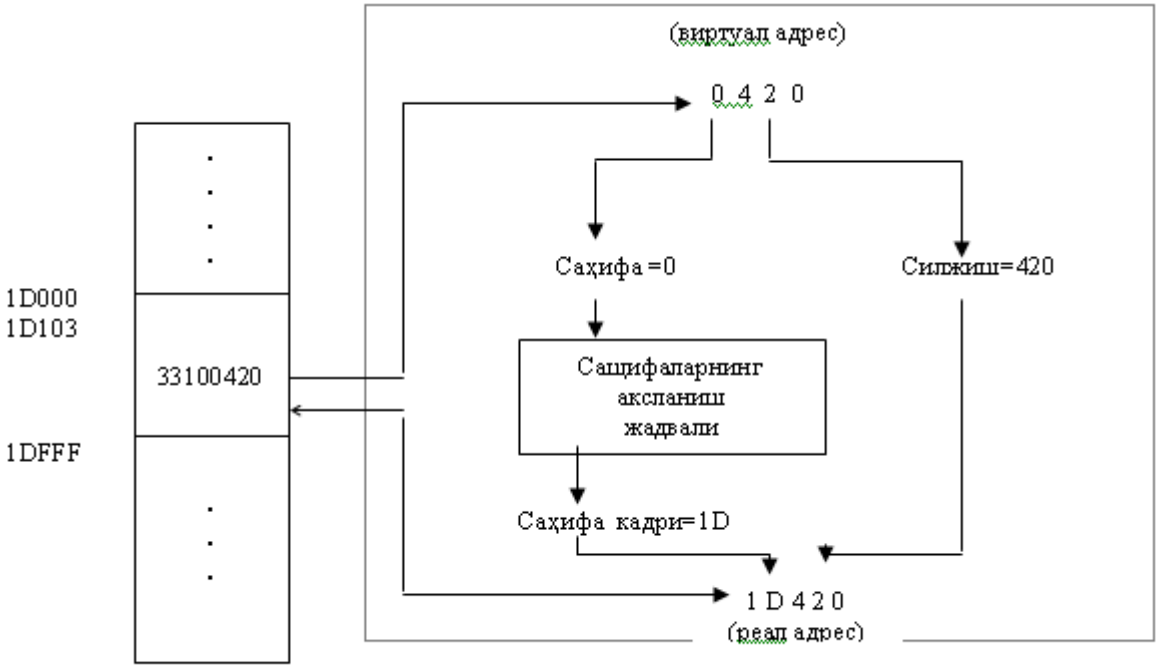


ВИРТУАЛ ХОТИРАНИНГ АСОСИЙ МОҲИЯТИ

Виртуал хотирадаги программа томонидан ишлатиладиган мълумотлар қандайдир ташқи қурилмада (ташқи хотирада) жойлашади. Зарурат туғилганда бу маълумотларниг бир қисми оператив хотирага кўчирилади. Ташқи хотира, унинг реал хотирага аксланиши ҳақида программа (фойдаланувчи) ҳеч нима билмайди. Программа шундай ёзилганки, унинг учун виртуал хотира амалда мавжуд хотира деб ҳисобланади. Ушбу маърузадан виртуал хотирани амалга оширишнинг умумий усулларидан бири –саҳифаларни сўров бўйича жойлаштириш усулини кўрамиз. Саҳифаларни сўров бўйича жойлаштириш тизимларида жараён виртуал хотираси қандайдир фиксирланган узунликдаги саҳифаларга бўлинади. Реал хотира ҳам худди шундай узунликдаги саҳифа кадрларга бўлинади. Потенциал равишда ихтиёрий жараённинг ихтиёрий саҳифаси реал хотира кадрига юкланиши мумкин.

Саҳифаларни кадрларга аксланиши саҳифаларнинг аксланиш жадвали (САЖ) ёрдамида амалга оширилади: тизим ҳар бир жараён учун битта САЖ ажратади. қурилмалар бу жадвалдан виртуал адресларни реал хотира адресларига алмаштиришда фойдаланади. Алмаштириш жараёни реал хотирани бошқаришдаги силжиш регистрини ишлатишга ўхшашдир. Виртуал адресларни реал адресларга бундай акслантиришни адресларни *динамик ўзгартириш* дейилади. *Мисол учун*, программа узунлиги 1000 (16 саноқ тизимида) байтли саҳифаларга бўлинган бўлсин. Программанинг 0-саҳифасига 0000 дан 0FFF гача виртуал адреслар тегишли, 1-саҳифага 1000 дан 1FFF гача адреслар тегишли ва ҳакоза. Программма бажарилиши бошланганда оператив хотирага биринчи бажарилувчи буйруқ бўлган 0-саҳифа қандайдир реал хотира кадрига юкланади. қолган саҳифалар зарурат бўлганда хотирага юкланади.

Адресларни динамик ўзгартириш ва саҳифаларни хотирага юклаш қуйидаги расмда келтирилган. Фараз қилайлик, 0-саҳифа реал хотиранинг 1D адресига юкланган бўлсин (реал хотиранинг 1D000-1DFFF адреси). 0103 адресдаги буйруқнинг операнд адреси 0420 бўлиб, у саҳифа ичида 420 силжиш билан жойлашган. Саҳифа 1D000 реал адресга эга бўлганлиги учун операнд адреси 1D420 бўлади.



Адресларни Динамик алмаштириш

Айрим тизимларда ҳар бир саҳифага қилинган охирги мурожат ёзиб қўйилади ва узоқ вақт ишлатилмаган саҳифа хотирадан чиқариб юборилади. Бошқа усул ҳар бир жараён учун у кўп ишлатиладиган саҳифалар тўплами аниқланади ва бу жараён актив пайтида хотирада айни шу саҳифалар тўплами бўлишига ҳаракат қилади. Саҳифалар билан ишлашда "бир жойда ҳаракатланиш" деб номланувчи ҳолат юзага келиши мумкин. *Мисол учун,* программа ишлашида ташқи хотирадаги саҳифага 100 марта мурожат бўлсин. У ҳолда марказий процессордан сўнг мурожаат 1мкс, ташқи ҳолатидан саҳифани ўқишига 10000 мкс кетади деб ҳисобласак, жараённинг 99% вақти саҳифалар устида ишлашга 1% фойдаланиш ишига кетар экан. Бу ҳолда қутилишнинг яна бир усули локал мурожат усулидир. Одатда программа кодида бажарилувчи буйруқларнинг гуруҳлашуви кузатилади. Бунга программада буйруқларнинг кетма-кет жойлшуви, ихчам цикллар, берилганлар структурасини кетма-кет қайта ишлаш орқали эришиш мумкин. ОС томонидан программани барча адрес соҳасини реал хотирага юкланмасдан, унинг маълум фрагментларини юклаш ва шу орқали минимал саҳифа узилишларига эришиш мумкин.

Умуман, оптимал ечим ҳар бир топшириқ учун ўзгариб туради. Лекин кўп масалалар учун қандайдир W критик нуқта мавжуд. Агар хотирада W миқдордан кам саҳифа бўлса «*бир жойда ҳаракатланиш*» ҳолати юзага келади. Саҳифаларнинг сўров бўйича жойлаштиришнинг бошқа усули кечиктирилган боғланиш усулидир виртуал ва реал хотира ўртасида боғланиш биринчи мурожаатгача ўрнатилмайди.

ВИРТУАЛ хотирани амалга оширишнинг бошқа усули хотиранинг сегментли ташкил қилишдир. Бунда адрес сегмент номери ва силжиш орқали аниқланади. Сегментларнинг реал хотирага акслантириш худди юқоридаги усулларда бўлади. Бироқ сегмент узунлиги ихтиёрий бўлиши ва одатда программа узунлиги билан мос тушади (ёки унинг блоклари ўлчамида). Бу ҳол сегментлар фақат ўқиш ёки фақат ёзиш каби ҳимоя атрибутларини қўллаш мумкин.

Назорат саволлари

1. Хотирани фиксирланган узунликдаги бўлаклар билан тақсимлашда ишлатилмай қоладиган хотира бўлакларини камайтириш учун нима қилиш керак?
2. Фиксирланган ва ўзгарувчан узунликдаги бўлаклар усулларини самарадорлик нуқтаи назаридан қиёслаш мумкинми?
3. Бўлакларни силжитиш усулида хотирадан самарали фойдаланиш ва программалар бажарилишининг умумий вақтини камайтириш ўртасида бғлиқлик борми?
4. Хотирадан бўлаклаб фойдаланишда ҳимоя қанадй амалга оширилади?
5. Саҳифа ва кадр тушунчалари ўртасидаги фарқ нимада?
6. Саҳифа узилиш қачон юзага келади?
7. Виртуал хотира ва оверлей тузилишили программалар билан ишлаш орасидаги ўхшашлик ва фарқ нимада?
8. Windows ОС 4 Гбайт оператив хотира билан ишлаши мумкин, лекин физик оператив хотира 128 Мбайт. Windows ОС ишлашини тушунтириб беринг.

12-мавзу. **Операцион тизимларнинг машинага боғлиқ бўлмаган хусусиятлари**

Асосий саволлар:

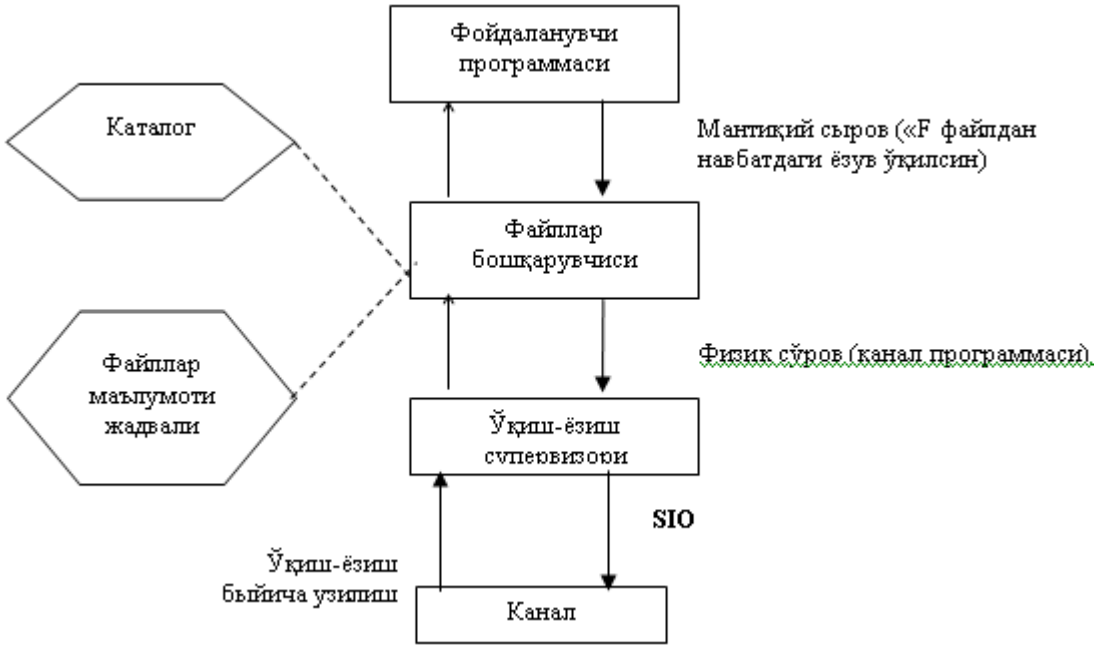
1. *Файллар билан ишлаш.*
2. *Топшириқларни режалаштириш.*
3. *Мультипрограммалик даражаси.*

Таянч ибора ва тушунчалар: *мантиқий ва физик сўровлар, мультипрограммалик, диспетчер, буфер, супервизор.*

Бў бўлимда ОС амал қилаётган машина архитектурасига боғлиқ бўлмаган айрим умумий функциялари қаралади. Бу функцияларни бошқа функцияларга нисбатан юқори босқичда амалда оширилиши мумкин.

Файллар билан ишлаш.

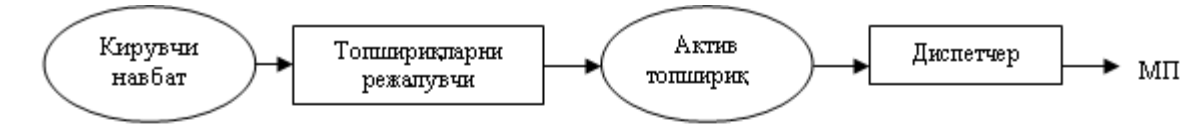
Кўпгина ОСларда ўқиш-ёзиш сўров канал программалари ва *SVC* сўров орқали амалга оширилади. Бу ҳолда программа тузувчи учун жуда ҳам ноқулай, чунки программа тузувчисига канал буйруқлари, улар фрматлари маълум бўлиши, программа учун канал ва қурилмалар номер ёзувлари реал адреслари ва ўқиш-ёзишни қандай кечишини кузатиш керак. ОСнинг файлларини бошқариш функцияси фойдаланувчи программаси ва ўқиш-ёзиш супервизори ўртасидаги оралиқ халқадир. Фойдаланувчи программаси мантиқий босқичда файл номи, калит ва бошқа воситалар ёрдамида «F файлдан навбатдаги ёзув ўқилсин» мазмунида сўров бериши мумкин.



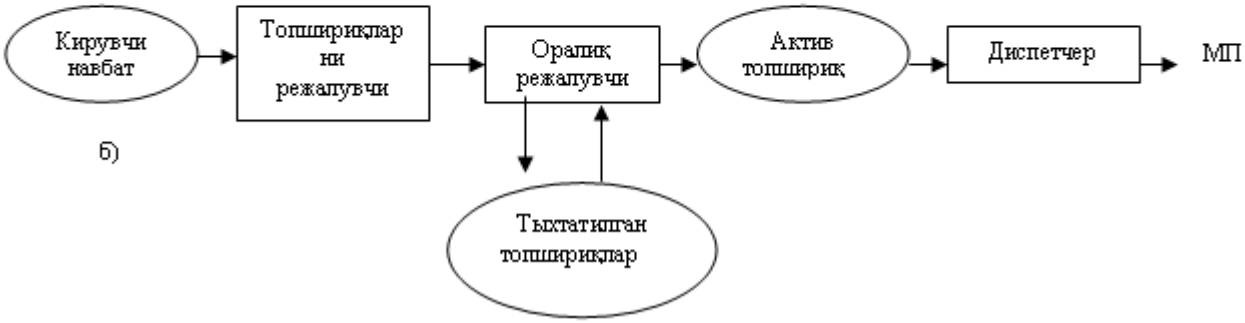
Файлларни бошқариш программаси мантиқий сўровларни физик сўровларга ўтказади ва уларни *супервизорга* узатади. Бунинг учун файлларни бошқариш тизими файллар ва уларнинг тузилишлари ҳақида маълумотларга эга бўлиши керак. Бу маълумотлар каталог ва файл маълумотлари жадвалидан олинади. Каталогларда файлнинг мантиқий номи билан унинг амалда жойлашган физик ўрни билан боғланиш ўрнатади ва файллар ҳақида бошқа умумий маълумотлар бўлади. Ўқиш-ёзиш бошланишидан олдин тизим файллар маълумот жадвалидан файл адресини аниқлайди. Бундан ташқари тизим файл блокларини ёзиш учун буферлар яратиши мумкин. Бу амалга (процедурага) *файларни очиш* дейилади. Файл билан иш тугагандан сўнг буфер ва бошқа ишчи саҳифаларга кўрсатгич йўқолади. Бу амалга *фални ёпилиши* дейилади. Файлларни бошқариш муҳим функциясидан бири ўқиладиган ва ёзиладиган файлларни автоматик равишда буферлаш ва блокларга бирлаштиришдир. Ўқиш-ёзиш учун файлларни бошқариш программасидан фойдаланиш қуйидаги расмда кўрсатилган. Фараз қилинадики, фойдаланувчи программаси файлнинг бошидан охиригача кетма-кет ўқийди. Мантиқий файл 1024 байтли ёзувдан иборат, амалда эса файл 8129 байтли блоклардан ташкил топган бўлиб, ҳар бир блок 8 та ёзувдан иборат бўлади. Берилганларни блокларга бирлаштириш операцион тизимлар томонидан қайта ишлаш ва хотирадан ютиш мақсадида амалга оширилади.

Юқоридаги расмда файл очилган фойдаланувчи программаси ёзувни ўқиш учун биринчи сўровни бергандаги ҳолат кўрсатилган. Бунда ОС буферга В1 блокни ўқишга буйруқ берган ва бу жараён тугашини кутаётган пайтда. В1 блок ўқилгач ва ОС фойдаланувчи программасига ёзувни ўқиш учун кўрсатгич программаси навбатдаги ёзувни ўқишга сўров берганда ўқиш-ёзиш учун бирор бир амал қилиш керак эмас. Фақат ОС Р кўрсатгични 2-ёзувга тўғирлаб қўяди. Фойдаланувчи программасида 9-ёзувга ўқиш бўлганда ОС В2 буферга 2 блокни ўқийди ва кўрсатгични 9 ёзувга тўғирлаб қўяди. Бундай усулга *жуфтли буферлаш* усули дейилади.

Топшириқларни режалаштириш. Мультипрограммалик ОСда икки босқичли режалаштиришга амал қилади: тизимга тушган топшириқлар кирувчи топшириқлар навбатига қўйилади ва кейинчалик улар топшириқлар режалаштирувчиси томонидан танланади. Танланган топшириқлар актив бўлади: улар жараёнларни режалаштириш амалида иштирок эта бошлайди. Агар марказий процессор ва бошқа ресурсларда бўш қолиш ҳолати юзага келса, топшириқлар режалаштирувчиси навбатдаги топшриқларни активлаштириш мумкин ва ЭҲМдан фойдаланиш самарадорлиги оширилади. Бироқ актив топшириқ кўплиги сабабли самарадорлик тушадиган бўлса, маълум бир топшириқлар тўхтатилиши керак. Бу вазифани оралиқ режалаштирувчи амалга оширади. Умуман олганда оралиқ режалаштирувчи мультипрограмма даражасини чеклаш мақсадида киритилган ва у ОСни маълум бир мультипрограммалик даражасида ушлаб туришга хизмат қилади. Лекин бу даража бажарилувчи топшириқ характерига боғлиқ равишда ўзгариши мумкин.



а)



б)

Юқорида келтирилган расмда икки кўринишдаги топшириқлар режалаштирувчилари келтирилган:

а) икки босқичли режалаштириш тизими;

б) уч босқичли режалаштириш тизими.

қайд қилинганидек, топшириқлар кўплигидан самарадорлик тушган бўлса, топшириқларни камайтириш қийинроқдир. Одатда бу муаммони ечиш учун босқичли режалаштириш қўлланилади. Унинг икки босқичликдан фарқи шундаки, режалаштиришда оралиқ режаловчи босқичнинг мавжудлигидир. Унинг вазифаси мультипрограммалик даражасини ўзгартиришдир. Бунинг учун айрим топшириқлар тўхтатилади, тўхтатилганлари ишга туширилади ёки топшириқлар режаловчисига навбатдаги топшириқларни активлаштириш учун мурожат қилинади. Режалаштириш усули маълум бир мақсадга эришиш учун мўлжалланган имтиёзлар тизимига таянади. Биринчи мақсад, тизимнинг максимал ўтказувчанлигига эришишдир, яъни қисқа вақтда иложи борича кўп масалани ечиш, бошқаси-топшириқнинг ўртача ўтиш (ишлаш) вақтини қисқартириш

(топшириқ юкланиши ва унинг тугаш вақтини энг кичик қилиш), учунчиси, жавоб вақтини қисқартириш: ENTER тугмасини босиш вақти билан прграмма жавоби вақтини камайтириш. Биринчи ва иккинчи шартлар ўзаро сиғиша олмайдиган мақсаддир. *Мисол* учун, вақт тақсимоти режимида катта сондаги терминлар билан ишловчи тизимни кўрайлик. Тизим самарали ишлаши учун, яъни тезкор жавоб бериш учун, бошқарувни бир терминалдан бошқа терминалга катта тезликда ўтиш етарли деб ҳисоблашимиз мумкин. Бунинг учун *диспетчер* ҳар бир жараёнга (терминалга) жуда кичик вақт кванти ажратади. Бироқ, бу ҳолда операцион тизимни топшириқдан топшириққа "кўчиш"га кўп вақт кетади, фойдаланувчи топшириғи учун эса кам вақт, шу сабабли тизим самарадорлиги умуман камаяди. Иккинчи мисол, пакетда бир программалик операцион тизимда иккита топшириқ ишлаётган бўлсин. Мультипрограмма режимида: а) биринчи ҳолда (бир программали операцион тизимда) марказий процессор бўш вақти анчагина; б) иккинчи ҳолда мультипрограмма ҳолида марказий процессор бўш вақти қисқарган. Шу сабабли иккита топшириқ тез бажарилади-5 минут ўрнига 4,5 минутда. Тизимнинг ўтказувчналиги ошди, бироқ топшириқнинг ўтиш вақти камаяди-3,5 минут ўрнига 4,4 имнут. Одатда жараёнларни режалаштиришда иккита усул қўлланилади: биринчи келганга биринчи хизмат; қисқа топшириқлар биринчи навбатда. Биринчи усулда барча топшириқларга бир хил хизмат қилинади, ўтиш вақти минималлаштирилади. Иккинчи усулда ўтишнинг ўртача вақтидан қисқаради.

Назорат саволлари

1. қандай қилиб мантиқий сўровга физик сўров мос қўйилади?
2. Файлдан ўқиш-ёзиш амали бажарилишида буфер ишлатилишининг бош сабаби нимада?
3. Мультипрограммалик даражаси деганда нима тушунилади?
4. Икки ва уч босқичли режалаштириш ўртасидаги фарқ нимада?
5. Топшириқларни режалаштириш критерияларини санаб беринг ва изоҳланг.

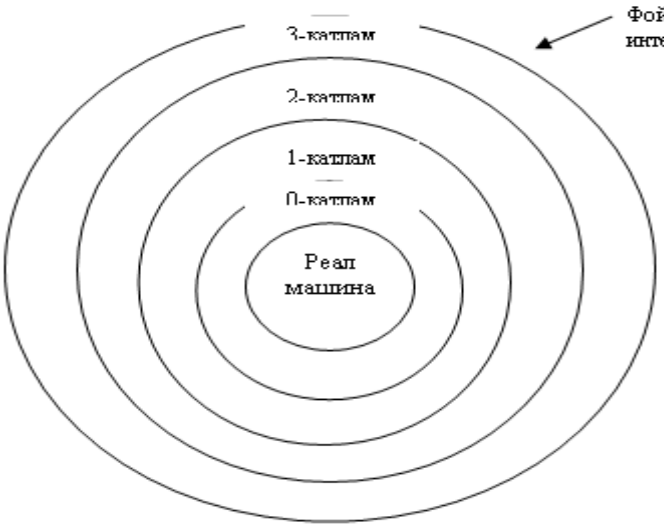
13-мавзу. **Операцион тизимларни қуриш усуллари**

Асосий саволлар:

1. *Шажаравий тузилиш.*
2. *Виртуал машина.*
3. *Мультипроцессорли тизимларни ташкил қилиш.*

Таянч ибора ва тушунчалар: *иерархик, интерфес, виртуал машина, SIO, STI, LPS, рессурслар, мультипрограммалик тизимлар.*

Иерархик (шажаравий) тузилиш. Реал ОТнинг анча қисми шажаравий тузилишга эга. Бундай тузилишга мисол қуйидаги расмда келтирилган:



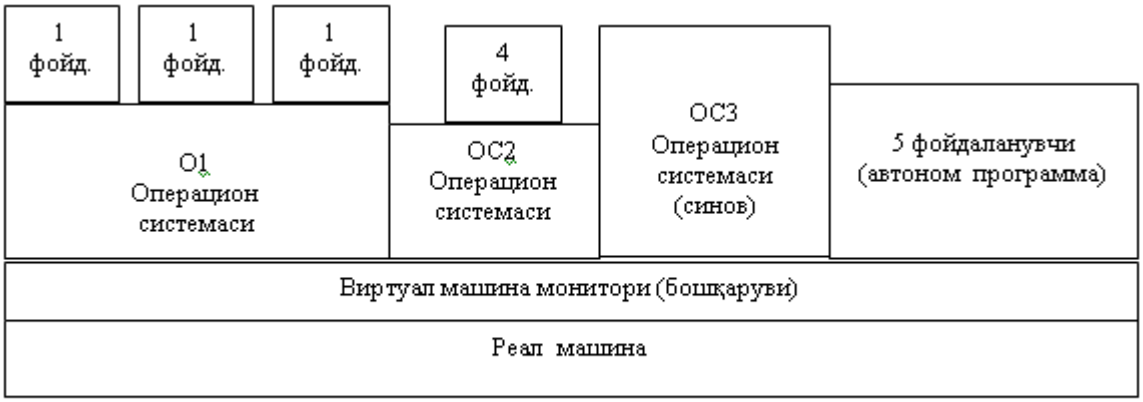
Қатлам	Вазифаси
3	файлларни бошқариш
2	хотирани бошқариш
1	Ўқиш-ёзишни бошқариш
0	Диспетчирлик ресурсларни бошқариш

Шажаравий тузилишли операцион тизим.

Ҳар бир қатлам (сатҳ) пастки қатлам томонидан бериладиган функциялардан фойдаланиш мумкин (бу ҳолда пастки қатлам реал машина қисми сифатида қаралади). 0-сатҳ бошқа ҳолда «ўзак» деб номланади, чунки бу сатҳ бевосита аппаратура билан иш олиб боради. 1 сатҳ 0-сатҳ томонидан берилган интерфес билан ишлайди. Фойдаланувчи программаси энг энг юқори сатҳ (3-қатлам) интерфейси билан иш олиб боради. Ҳар бир сатҳда шу сатҳдаги ва ундан пастдаги функцияларга мурожаат қилиши мумкин. Бизнинг мисолда 3-қатламда файлларни бошқаришда буферни тақсимлаш учун 2-қатламга мурожаат қилиши, ўқиш-ёзиш учун 1-қатламга тегишли ўқиш-ёзиш супервизорига мурожаат қила олади. Барча қатламлар 0-қатламдаги жараёнларни режалаш ва ресурсларни бошқариш функцияларидан фойдаланиши мумкин.

Иерархик ОС маълум бир афзалликларга эга. Ҳар бир қатлам ўзидан пастки қатлам томонидан бериладиган, нисбатан содда функция ва интерфейсларнигина ишлатиши мумкин. Программа тузувчи учун у фойдаланётган функциялари қандай амалга оширганлигини билиши, унинг моҳиятига етиши шарт эмас. ОС 0-қатламдан бошлаб яратилиши ёки созланиши мумкин, яъни қатлам бўйича юқоридаги расмда келтирилган. ОС қатламларининг функциялари жойлашуви қатъий эмас. Айрим ОСларда 1-даражали узилишларни қайта ишловчи (FLIH) 0-қатламда (ўзакда) жойлашган. Узилишни бошланғич равишда қайта ишлагандан кейин бошқарувни юқори қатламдан прогрраммага узатиш мумкин. Бу ҳол юқори қатламга мурожаатни ман қилувчи қоидадан четланишдир. Иерархик ОСларда пастки қатламларга босқичма-босқич мурожаат қилиш қабул қилинган. Айрим ҳолларда бу операцион тизимларнинг самарадорлаигини пасайишига олиб келади. Шу сабабали *шафоф операцион* тизимларда қатламдан пастки қатламларга тўғридан-тўғри мурожаатга рухсат берилган.

Виртуал машина. Фойдаланувчиларнинг ҳар бири ўзини автоном виртуал машинададек ҳис қилиши учун иерархик ОС тушунчасини кенгайтириш мумкин. Бу усул турли операцион тизимни битта реал машинада бир вақтда ишлашга имкон беради.

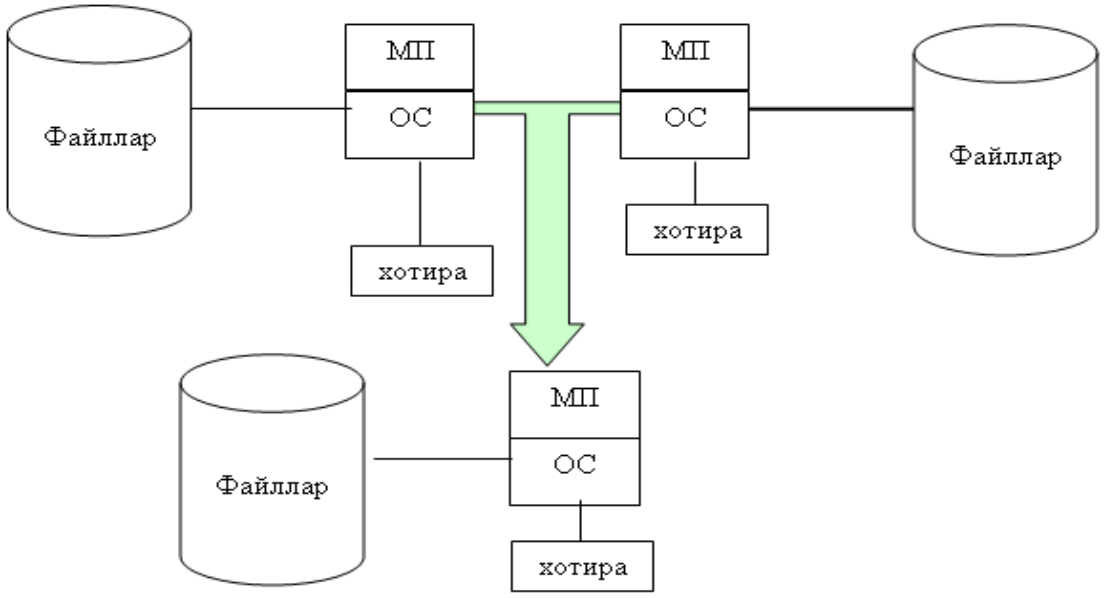


Виртуал машинали ва кўп фойдаланувчи операцион тизим.

Мисол учун, ОС1-мультипрограммалик ОС (учта фойдаланувчи), ОС2-бир прогаммалик ОС; ОС3-айни пайтда текширилаётган ОС. Булардан ташқари, 5-фойдаланувчи бўлиб, у супервизор режимида автоном программа сифатида ОС бошқарувидан ташқарида ишлайди. Буларнинг ҳаммаси битта *реал* машинада ишлайди. Бироқ улар реал машина билан эмас, балки виртуал машина монитори билан ишлайди ва улар бир-бирига боғлиқ бўлмайди. Биророта ОС ёки фойдаланувчи тизимга «шикаст» етказса, бу ҳолат фақат уларнинг автоном машинасига таъллуқли бўлади, бошқа ОСга таъсири бўлмайди. Виртуал машинанинг ҳар бири бевосита фойдаланувчи, мисол учун ОС1 ёки 5-фйдаланувчи, амалда супервизор режимида эмас, балки фойдаланувчи режимида ишлайди. *Мисол учун*, улар томонидан имтиёзли буйруқлар (*SIO,STI,LPS*) бажаришга уринганда, узилиш рўй беради ва бошқарув *ВММ*га узатилади. У ўз навбатида буйруқ бажарилишини «иммитация» қилади ва бошқарувни фойдаланувчига қайтаради. *ВММ* реал машина узилишидан ҳам активлашади. Узилиш қайси вирутал машина амал қилиши кераклигини аниқлайди ва бу виртуал машина ҳолатига ўзгартириш қилади. Амалда *ВММ* реал машинанинг содда ва тугалланган операцион тизимидир.

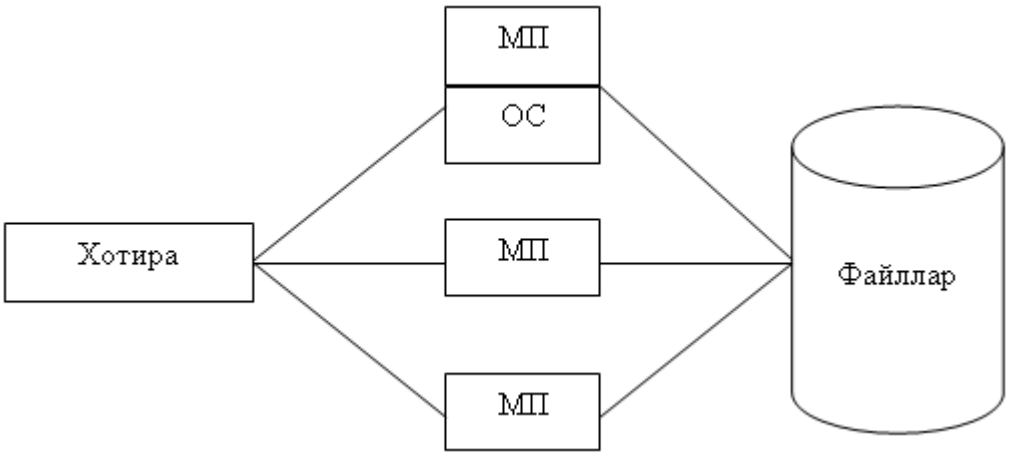
Вирутал машинанинг афзалликларидан энг муҳмми, унинг ўнғайлиги ва мосланувчанлигидир. Ҳар хил тоифали фойдаланувчи талабини қондириш учун бир юқори даражадаги ҳимояга эришиш мумкин, чунки бир виртуал машина иккинчи виртуал машина ресурсларига мурожаат қила олмайди.

Мультипроцессорли тизимлар. Кўп процессорли ОСли машина бир процессорли тизимлардек амал қилиши керак. Фарқи шундан иборатки, жараёнлар режалаштирувичиси фойдаланувчи топшириқлари учун биттадан марказий процессор ажратиши, бир пайтда бир нечта жараён актив бўлиши мумкин. қуйидаги расмда мультипроцессорли тизимда ҳар бир процессор ўз хотирасига, ўқиш-ёзиш ва бошқа ресурсларга эга. Бундан ташқари ҳар бир марказий процессор ўзининг ОСга амал қилади. Ҳар бир процессор бошқаси билан алоқа сими орқали боғланган. Бу сим орқали бир процессор иккинчисига сўров юборади ва бошқа процессор ресурсларига мурожаат бўлади. Бу усулга суст боғланган процессорли *мультипроцессор тизим* дейилади ва у бир процессорли тизимлар тормоғига ўхшатиш мумкин. Бу усулдан процессорлар ўртасида махсуслаштириш функцияси мавжуд бўлганда мақсадга мувофиқдир. *Мисол учун,* вақт тақсимоти режимида ишлайдиган тизимларда, ҳар бир терминал учун ўзининг процессори мавжуд бўлсин, асосий процесоор эса барча ҳисоблаш ишларини бажаради (зарур бўлганда барча терминал процессорлари билан баоғланган ҳолда).



Процессорлари суст боғлиқ бўлган мультипроцессорли тизим.

Суст боғланган процессорли мультипроцессор тизимлар нисбатан содда, чунки ҳар бир ресур қандайдир битта процессорга ажратилади. Иккинчи томондан, ресурслар процессорлар ўртасида тўғридан-тўғри тақсимланиши мумкин. Бу ҳолга *кучли боғланган* процессорли мультипроцессор тизимлар дейилади ва улар маълум маънода ОС вазифасини қийинлаштиради. Мультипроцессорли тизимларни ташкил қилишнинг яна бири етакчи-қарам усулидир. Ресурсларни ва ОС нинг бошқа функцияларини битта асосий процессор томонидан бошқарилади. Процессорларнинг ўзаро алоқаси махсус алоқа симлари орқали ёки биргаликда ишлатиладиган хотиранинг ишчи соҳалари орқали ўрнатилади. Процессорлар хотира ва берилганлар, файллар биргаликда ишлатилиши мумкин, лекин ОС ташкил қилувчи программалари ва берилганлар структураси фақат асосий процессор томонидан фойдаланилади. ОСнинг бундай тузилиши маълум бир камчиликларга эга, асосий процессорларнинг ОС хизмати билан банд бўлиб қолганда бошқа процессорлар «туриб» қолиши мумкин, иккинчидан асосий процессор ичида юз бериши мумкин. Бу муаммони ечиш учун ҳар бир процессорга ОС ва фойдаланувчи томонидан талаб қилинадиган функцияларни бажаришга рухсат беришдир.



Етакчи- қарам кўринишдаги мультипроцесорли тизим.

Бу усулга *симметрик қайта ишлаш тизимлар* дейилади. Симметрик тизимларда ОСнинг турли қисмлари бир вақтда амал қилиш мумкин. Шу сабабли бундай операцион тизимни амалга ошириш бошқаларга нисбатан мураккаб бўлади. Барча процессорлар ОС томонидан ишлатиладиган берилганларга мурожаат қила олади. Бунда иккита процессорлар битта ресурсларни ишлатишда ўзаро келиша олмаслиги мумкин. Бу муаммони ечиш учун ОС ресурсларга мурожаат қилишнинг махсус воситаларидан ва критик ҳолат жадвалидан фойдаланиши мумкин.

Назорат саволлари

1. ОСни шажаравий усулда қуришнинг афзалликлари нимада?
2. Виртуал машина моҳиятини тушунтиринг.
3. Мультипрограммалик тизимлар ташкил қилиниш усуллари бир-биридан принципиал фарқи нимада?
4. Windows ОС қайси турдаги ОС киради?

14-мавзу. **MS DOSнинг тузилиши ва функциялари.**

Асосий саволлар:

1. *Операцион тизимлар структураси ва тарихи.*
2. *MS DOSнинг асосий модуллари.*
3. *BIOSнинг асосий вазифалари.*
4. *Autoexec.bat файлининг тузилиши.*

Таянч ибора ва тушунчалар: *модуллар, утилиталар, сектор, йўлак, узилиш.*

Операцион тизимлар структураси ва тарихи. Операцион тизимлар структурасига кўра бир неча версиялари мавжуд:

Операцион тизимлар
CP/M-80 (8 разрядли ШК лар учун)
CR/M (16 разрядли ШК учун)
MS DOS (64 Кб хотирага эга былган ШК ларг учун)
MS DOS 1.1.,1.25,2.0,2.11 ва х.к.
MS DOS 3.3 (640 Кб хотирага эга былган компьтерлар)
PC DOS
DRD DOS
OS/2
WARP
UNIX
Windows
Linux
ва щ.к.

IBM фирмаси 1986 йилда 80386 микропроцессорига асосланган компьютерни ишлаб чиқарди. Бу микропроцессор асосида яратилган компьютер назарий бир неча Гегабайт хотирага эга бўлиши мумкин эди. Аммо MS DOS ОС 640 Кб бўлган компютерларга мослашган эди. Шунинг учун MS DOS ОС ни кенгайтириш мақсад қилиб олинди ва 1987 йил MS DOS 3.3 версияси яратилди. Шу йили Microsoft фирмаси томонидан бир вақтда бир нечта масалалар ечишга қодир бўлган OS/2 ОС ишлаб чиқилди. Лекин бу ОС кенг тарқалмади, сабаби MS DOS 3.3 кўпчилик имкониятларини қониқтирар эди. Хозирга келиб ОС ларнинг бир неча турлари ишлаб чиқилган ва ишлаб

чиқилмоқда. Аммо MS DOS ҳалида ўз кучини йўқотгани йўқ. MS DOS нинг қобиқ дастури бу NC ҳисобланади.

ОС функциялари. ОС-бу бошқарув дастуридир. ОС бу компьютернинг физик ва дастурий ресурсларини тақсимлаш ва уларни бошқариш учун ишлатиладиган дастур. Компьютер ресурслари икки хил бўлади: *физик* ва *дастурий.*

Физик ресурслар бу:

-хотира;
-винчестер;
-монитор;
-ташқи қурилма.

Дастурий ресурслар бу:

-киритиш ва чиқаришни бошқарувчи дастурлар;
-компьютер ишлашини таъминлайдиган бошқарувчи дастурлар;
-берилганларни таҳлил қилувчи дастурлар;
-драйверлар;
-виртуал ички ва ташқи хотирани ташкил қилувчи ва бошқарувчи дастурлар.

ОСлардан қуйидаги хусусиятларга эга бўлиши талаб қилинади:

1. **Ишончлилик.** ОС ўзи ишлаётган қурилмалар билан бирга ишончли бўлиши керак.ОС фйдаланувчининг айби билан вужудга келган хатони аниқлаши, таҳлили қилиши ва тиклаш имкониятига эга бўлиши керак.
2. **Ҳимоя.** ОС бажарилаётган масалаларнинг ўзаро бир-бирига берадиган таъсиридна ҳимоялаш керак.
3. **Башорат.** ОС фойдаланувчи сўровига башоратчилик билан жавоб бериши керак.
4. **қулайлик.** Фойдаланувчига ОС ни таклиф қилишдан мақсад ресурсларни аниқлаш ва бу ресурсларни бошқариш масалалалрини есишдан озод қилишдир. Системани инсон психологиясини ҳисобга олган ҳолда лойиҳалаш керак.
5. **Эффективлик.** Ресурслар тақсимотида ОСТ фойдаланувчи учун максимал ҳолда система ресурсларидан фойдаланиш даражасини ошириш керак. Ресурсларнинг ОС томонидан банд қилиниши фйдаланувчи имкониятларини камайтиришга олиб келади.
6. **Мосланувчанлик.** Система амаллари фойдаланувчига қараб созланиши мумкин. Ресурслар мажмуаси ОС эффективлиги ва самарадорлигини ошириш мақсадиди кўпайтириш ёки камайтирилиши мумкин.
7. **Кенгайтирувчанлик.** ОС га янги физик ва дастурий ресурслар қўшилиши мумкинлиги.
8. **Аниқлик.** Фойдаланувчи система ҳақида қанча билгиси келса шунча билиш имкониятига эга бўлиши керак. ОС фойдаланувчи ресурслар тақсимотидан озод қилиб компьютерни уч хил режимда ишлашини таъминлаш мумкин: бир дастурли, кўп дастурли, кўп масалали.
9. **Бир дастурли режим.** Компьютернинг барча ресурслари фақат бир дастурга хизмат қилади.
10. **Кўп масалали режим.** Бу бир вақнинг ўзида бир неча масаланинг параллел ишлашини таъминлаш кўзда тутилган. Бунда бир масаланинг натижаси иккинч масала учун берилганлар мажмуасини ташкил қилиши ҳам мумкин. ОС ечилаётган масалаларнинг бир-бири билан боғлиқлигини режалаштиради ва назорат қилиб боради. Кўп масалали режим фақат мультисистемада ташкил қилинади.

MS DOSнинг асосий хусусиятларидан бири унинг модул кўринишида ташкил қилинганидаир. Бу хусусият, биринчидан мураккаб программалар мажмуаси ҳисабланган DOSни ўзаро боғлиқ бўлмаган бўлакларга ажратишг, иккинчидан ҳар бир модулга

мантиқий боғланган функциялар гурухини бирлаштиришга имкон беради. Агар DOS бирорта функциясига ўзгартириш керак бўлса, фақат мос модулга ўзгартириш киритилади.

MS DOS қуйидаги асосий асосий модуллардан иборат:

-ўқиш/ёзиш база тизими (BIOS).

-бошланғич юклаш блоки (BOOT RECORD)/

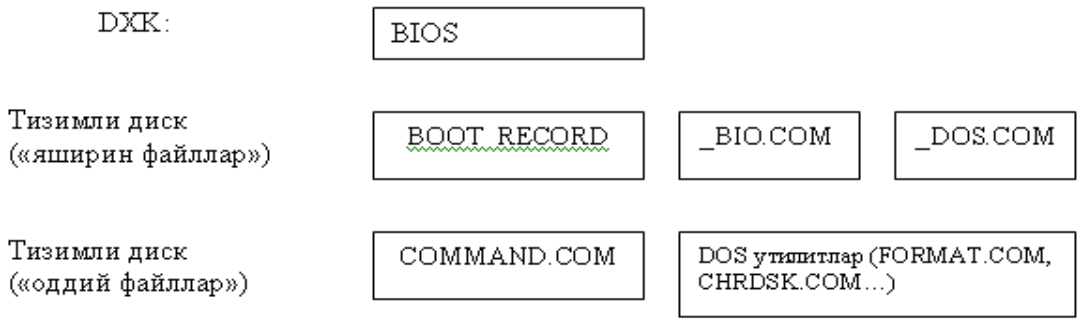
-ўқиш/ёзиш база тизимининг кенгайтма модули (BIO.COM ёки IO.SYS);

-узилишларни қайта ишлаш моули (DOS.COM ёки MSDOS.SYS);

-буйруқлар процессори (COMMAND.COM);

-DOS утилиталари (FORMAT.COM, CHRDSK.COM ва бошқа программалр).

Бу модулларнинг дискда жойлашуви қуйидагича:



BIOS хотирада доимий равишда бўлади ва у компьютер қурилмасининг бир қисми сифатида қаралади.

BIOSнинг асосий вазифалари:

1.ЭҲМ ишга тушганда асосий қурилмаларни автоматик равишда текшириш.

2. DOS бошланғич юклаш блокини чақириш. DOS нисбатан катта бўлганлиги учун у икки босқичда хотирага юкланади: олдин бошланғич юклаш блоки юкланади ва у ўз навбатида DOSнинг бошқа модулларини юклайди. BIOS нормал ишлаши учун асосий шартлардан бири-дискнинг фиксирланган жойидан *бошланғич* юклаш модулини топишдир. У олдин эгилувчан дискдан (A:) излайди, топилса, A: дискетаси тизимли деб қилинади, акс ҳолда «қаттиқ» дискдан изланади.

4. BIOSнинг учунчи муҳим вазифаси тизимли сўровлар ёки узилишларни қайта ишлашдир. Бу узилишлар уч тоифага бўлинади. Аппарат узилишлар-электр токи камайиши, клавиатура тугмасини босиш, таймер ва ташқи қурилмалар сигналлари. Мантиқий ёки процессор узилишлари-микропрцессор ишлашида юзага келадиган ностандарт ҳолатларга мос сигналлар-нолга бўлиш, регистрлар тўлиб кетиши.

Программа узилишлари- программа томонидан бошқа программанинг хизматидан фойдаланиш ҳолларида юзага келади. Одатда бу хизмат аппарат қурилмалари билан боғлиқ бўлади.

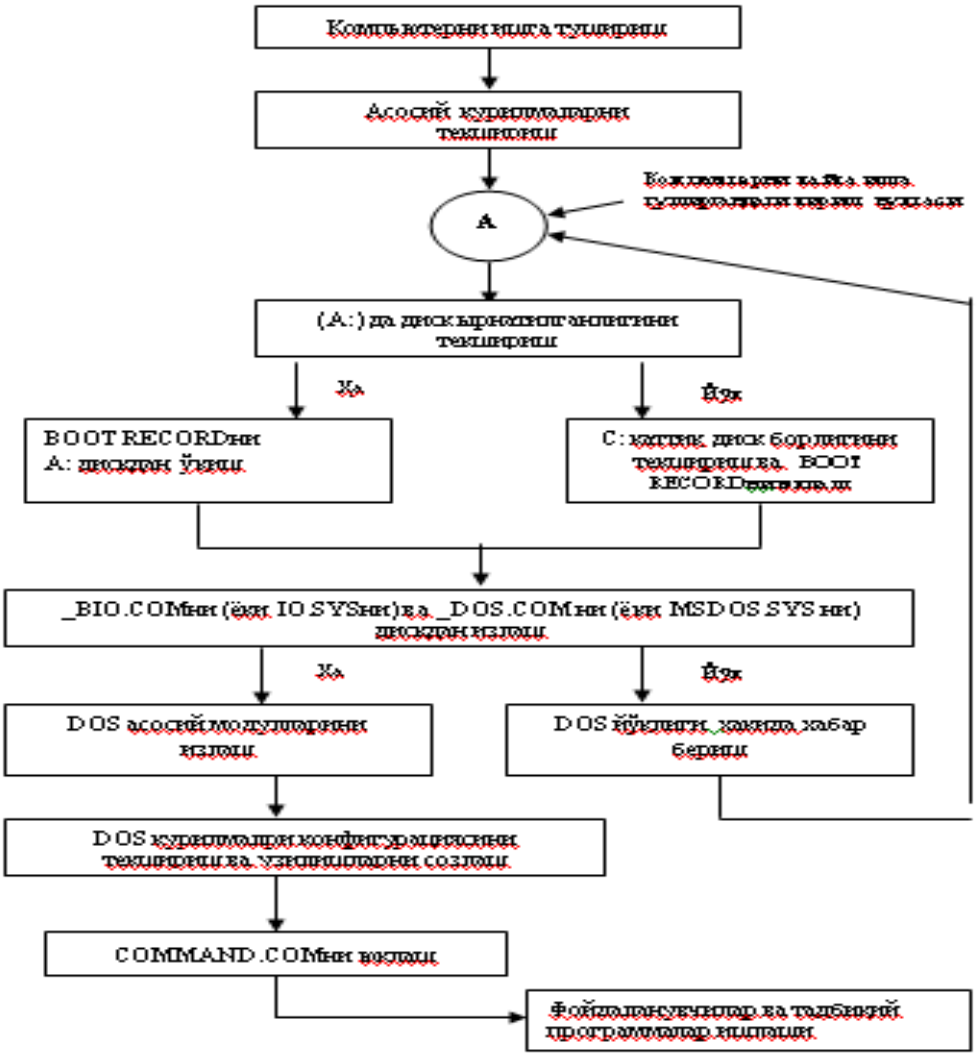
Бошланғич юклаш блоки- DOS иккита блокларини-BIOS кенгаймаси модули ва узилишларни қайта-ишлаш блокини юкловчи кичик программадир. Бошланғич юклаш блоки тизимли дискнинг 0-томон, 0-йўлакнинг 1-секторида жойлашади ва унинг узунлиги 512 байт бўлади.

Буйруқ процессори. Тизимли дискдаги учинчи модул-буйруқлар процессоридир (COMMAND.COM). Бу модул дискнинг ихтиёрий жойида жойлашиши мумкин ва оддий программа ҳисобланади. Унинг функциялари қуйидагилардан иборат:

1. *Клавиатура ва буйруқ файлидан киритилган буйруқни қабул қилиш ва синтактик анализ қилиш.*
2. *ОТ ички буйруқларини бажариш.*

3. ОТ ташқи буйруқларини ва фойдаланувчининг амалий дастурлаини юклаш ва бажариш.

Буйруқ процессори томонидан бажариладиган буйруқлар *ички буйруқлар* дейилади. Фойдаланувчи томонидан бажариладиган буйруқлар *ташқи буйруқларни* ташкил этади. ОС ташқи буйруқлари дискда алоҳида сақланган дастурлар ёрдамида бажарилади. Ихтиёрий ОС га турли амалларни бажаришга мўлжалланган ўнлаб дастурлар киритилган. *Масалан,* барча ОСларга киритилган *қурилма драйвери* деб номланадиган махсус резедент дастурлар киритиш-чиқариш системасини тўлдириш учун қўлланилади. Ушбу драйверлар қўшимча ташқи қурилмаларни ёки мавжуд қурилмаларни ностандарт ишлатилишини таъминлаб беради. қуйидаги расмда DOS нинг юкланишининг схемаси берилган:



DOSни юклаш ва инициализация қилиш

CONFIG.SYS файилининг тузилиши

Операцион тизим юкланадиган дискнинг ўзак каталогида CONFIG.SYS файли жойлашади. Бу файл DOS операцион тизимининг параметрларини беради ҳамда операцион тизимнинг имкониятларини кенгайтирувчи қайси программаларни оператив хотирага юклаш кераклигини кўрсатади (бу программаларга қурилмалар драйверлари киради).

CONFIG.SYS файлига мисол ккелтирамиз:

```
Break       =on
Files       =20
Buffers/ 16
Shell       = command.com /e : 512 / p
Country     = 007, 866, C: /EXE/MSDOS/COUNTRY.
Device      = C: /EXE/ SYS/ MOUSE . SYS 2
Install     = C: / EXE / RK.COM
```

CONFIG.SYS файилининг таркибига кирувчи буйруқларнинг вазифаларини кўрсатиб ўтамиз:

Break = on - дискдаги киритиш-чиқариш операцияларини бажаришда Ctrl/Break тугмачаларини босишни текшириш ҳолатини белгилайди. Бу программани тугаллангунча бажарилишини тўхтатади.

Files = 20 - Бир вақтда очиладиган файлларнинг максимал сонини белгилайди. Баъзи маълумотлар омбори билан ишлаганда параметрларнинг қийматлари Files: 50 ёки ҳаттоки 80 бўлиши зарур.

Buffers = буферлар-сони - дисклардаги киритиш-чиқариш операциялари учун *буферлар* сонини белгилаш. 20 Мбайтли қаттиқ дисклар учун 15-20 буферлар, 20-40 Мбайтли дисклар учун эса 30-40 буферлар, 40 Мбайтдан юқори дисклар учун 40 буферлар ишлатилади. қаттиқ дискларни кешлашда *буферлар* сонини минималга келтириш мумкин (4-5).

Shell= command.com/e: 512/ p - Тизимнинг ўзгарувчилари сақланадиган хотира соҳаси ўлчамини кенгайтиради. Байтлар сони бу соҳанинг ўлчамини ҳосил қилади. Агар хотира соҳасининг ўлчами тизим ўзгарувчиларини сақлаш учун етарли бўлмаса, у ҳолда DOS «Out of environment space» номли хабарни беради.

Country= 007, 866, C:/EXE/MSDOS/COUNTRY - давлатнинг коди, вақт ва сана ҳақидаги маълумотни беришнинг қулай формати. MS DOSда 6.2 версиясидан бошлаб қуйидагилар Россия учун аҳамиятга олинган: давлат коди 007, кодли саҳифа 866.

Device=C:/EXE/SYS /MOUSE.SYS 2 - қурилмаларнинг драйверлари ўрнатилади. Драйвер-программалар тизимнинг имкониятларини кенгайтиради. Энг муҳим *драйверларга* миоллар келтирамиз:

ANSI.SYS - клавиатурадаги тугмачаларнинг вазифасини қайта аниқлайди ва экранга маълумотларни чиқариш имкониятини кенгайтиради;

DRIVER.SYS - компьютерга қўшимча дискларни улайди;

MOUSE.COM- амалий программаларда сичқончадан фойдаланишни таъминлайди;

SMARTDRV.EXE - дискни *кэшлаш*ни таъминлайди;

RAMDRIVE.SYS - кенгайтирилган ёки қўшимча хотирада *электрон дискни* яратади.

Install =C:/EXE/RK.COM -резидент программани ўрнатиш. Бу усул оператив хотирани эканом қилади. Юкланувчи программа учун ундан фойдаланганда, муҳитнинг ўзгарувчиларини сақлаш учун хотирада жой ажартилмайди.

AUTOEXEC.BAT файлининг тузилиши

DOS операцион системаси юкланишида дискнинг ўзак каталогда **AUTOEXEC.BAT** файлини излайди. Агар бу файл топилса у бажарилади. **AUTOEXEC.BAT** буйруқ файлига операцион тизимни ҳар сафар юкланишида бажариладиган буйруқларни ёзиш қулай. Бу буйруқлар операцион тизимнинг керакли созлашларини амалга ошириб, қулай муҳитнинг ишлашини яратади. *Маълумки*, DOSнинг *Path* буйруғи ёрдамида тизим юкланишида бажариладиган дастурларни қайси

каталоглардан излаш кераклигини кўрсатиш мумкин. Одатда бу буйруқ **AUTOEXEC.BAT** фалига киитилади. *Path* буйруғида бажарилувчи дастурлар жойлашган каталоглар рўйхатини *нуқта-вергул* ва *“/”* белгиси билан ажратиш зарур. *Path* буйруғининг ёзилишига қуйидаги мисолни келтирамиз:

Мисол: path/c:/exe;c:/exe/dos;c:/exe/nu;c:/tc;…;../..

AUTOEXEC.BAT файлига мисол келтирамиз:

```
@echo off
rem       Дастурни излаш йўлини кўрстиш
path c:/exe/Dos;c:/exe;c:/exe/nu;c:/ғtc;…;../..
rem       DOS таклифини кўринишини белгилаш
promt $p$q
set  TEMP/=D:/WIN/TMP
rem                 Клавиатура драйверини ишга тушириш
C:/EXE/FILOAD/RK
rem                 DOSKEY  программасини ишга тушириш
DOSKEY
rem                 файллардаги  ўзгаришларни текшириш (вирусдан ҳимоялаш)
rem
c:/exe/adinf/ғadinf.exe –a  -b  - d  -s  -u  -lc:/exe/adinf  c:  d:  e:
rem                 вируслардан ҳимоялаш дастурларини ишга тушириш
c:/exe/-d.com   -V   -R
rem                 Norton Commander дастурини ишга тушириш
NC
```

Дискларни КЭШлаш

Кўп программалар ишлаётганда қаттиқ диск билан тинимсиз маълумотлар алмашадилар. Сиз уларнинг бажарилишини дискларни *кешлаш* ҳисобига етарлича тезлаштиришингиз мумкин. Дискларни кешлаш программалари компьютернинг тезкор хотирасида дискнинг энг кўп ишлатиладиган қисмига тез мурожатни таъминлайдиган *буфер* яратади. Бунинг натижасида киритиш-чиқариш операциялари тезлиги бер неча марта ошади ва ортиқча юкларнинг бундай камайиши эса *дискларнинг* хизмат қилиш муддатини янада узайтиради. Дискларни кэшлаш программалари DOS такибига киради. Шунингдек турли фирмалар томонидан етказиб берилади. *Масалан,* MS DOS га кирувчи SmartDriver програмаси (SMARTDRV.EXE), Norton Utilities тўпламига кирувчи – Norton Cache (NCACHE2.EXE) ва ҳ.к. Windows операцион тизими ўрнатилишида CONFIG.SYS файлида SMARTDRV.EXE драйверини ишга туширувчи *сатрни* ҳосил қилади. Дискларни кэшлаш- компьютернинг иш унумдорлигини оширувчи самарали усулларидан биридир. Кэшнинг ўлчами программанинг бажарилиши учун тезкор хотирада қанча жой ажратилишига боғлиқ. Дискларни кэшловчи барча программалар DOS ва Windows билан ишлаганда *кэшнинг* ўлчамини алоҳида кўрсатади. *Масалан,* MS DOS таркибига кирувчи SMARTDRV программаларни ишга тушириш учун CONFIG.SYS файлига қуйидаги сатрни киритиш керак.

Масалан. DeviceқC:/EXE/MSDOS/SMARTDRV.EXE 2048 1536.

Бу ерда 2048 Кбайт (2 Мбайт) DOS учун кэшнинг ўлчами, 1536 Кбайт (1,5 Мбайт) Windows учун кэшнинг ўлчами.

15-мавзу. **Windows операцион тизими**

Асосий саволлар:

1. *Windows ҳақида умумий тушунчалар;*
2. *Windowsдаги виртуал машиналар;*
3. *Windowsда оқим тушунчаси;*
4. *Диспетчер ишлайдиган алгоритмлар.*

Таянч ибора ва тушунчалар: *оқим, диспетчер, приоритет, Win32, Win16.*

Ҳозирги пайтда Windows шахсий компьютерлардан энг кўп тарқалган операцион тизимдир: дунёда 150 млн.дан кўп IBM PC тоифасидаги компьютерлар мавжуд бўлган ҳолда, уларнинг 100 млн. ортиғи Windows ўрнатилган (1997 й). 1985 йилда Microsoft фирмаси томонидан илк бор Microsoft Windows 1.0 операцион қобиғи тақдим этилди. Кенгайтирилган хотирани бошқариш протоколини қўллаб-қувватловчи 80386 процессорларининг юзага келиши Windows/386 операцион қобиқ яратилишига сабаб бўлди. Бу муҳитда қўшимча хотира ҳисобига хотирани кенгайтириш, виртуал режимда бир нечта программани бир вақтда бажариш имконияти юзага келди. 1990 йилда Windows 3.0 версиясининг яратилиши олдинги қўйилган катта қадамлардан бири бўлди. Microsoft фирмаси 80286 ва 80386 процессорлар учун ҳимояланган режимни киритиш натижасида тадбиқий программалар учун 16Мб хотирагача ажратиш имкони пайдо бўлди (хотира саҳифа кўринишида эмас, яхлит ҳолда ишлатилади). Кўппрограммалик режими амалга оширилади ва DOS программаларини алоҳида ойна (windows) ичида бажариш мумкин.

Windowsнинг аввалги 3.0, 3.1, 3.11, 3.12 лар асос сифатида MS DOSни қабул қилган бўлса, ҳозирда мустақил бошқа операцион тизимни бўлишини талаб қилмайди. Лекин шу билан бирга бу муҳитда MS DOS ва Windowsнинг эски кўринишлари, яъни версиялари ишлаш имконияти сақланган.

У қуйидаги афзалликларга эга:

- *ўзлаштириш ниҳоятда оддий ва имкониятларидан фойдаланиш имкони мавжуд;*
- *юқори самарадорликка эга ва аввалги версияларидан кескин фарқланиши;*
- *фойдаланувчи битта дастурий таъминот остида бир неча имкониятга эга бўлишлиги;*
- *юқори ва тушунарли имкониятлари мавжудлиги.*

1995 йил Windows 95 операцион тизими сотувга чиқарилди. Энди фойдаланувчилар объектга йўналтирилган интерфейсга эга бўлишди: туб маънода «иш столи» ва пиктограмма «олиб ўтиш» техникаси ёрдамида нусха олиш ва ўчириш, ичма-ич жойлашган папкалар ва ҳоссаларни бериш учун енгил диалог воситалари, узун номли файлларни ишлатиш имкониятлари пайдо бўлди. Windows 95 операцион тизими Windows архитектурасини сезиларли даражада яхшилади. *Жумладан*, Windows 95 ОТда 32 разрядли тадбиқий программалар интерфейси (API), 32 разрядли тадбиқий программалар учун ҳимояланган адрес фазоси, тадбиқий программаларни оқимларга бўлиш ва қурилмаларни виртуал драйверларидан янада кенгроқ фойдаланиш амалга оширилди. Шу билан биргаликда 16 разрядли тадбиқий программаларни ишлатиш имконияти сақлаб қолинди. Windows 95 ОТда 32 разрядли хотира ўлчамини 4 Гбайтгача оширишга имкон беради. Амалда 4 Гбайт оператив хотира мавжуд эмас. Windows *виртуал машиналар диспетчери* (VMM) бу ўлчамдаги хотирани мавҳум ҳолда ташқи хотирада ҳосил қилиб, зарур бўлганда оператив хотирага юклашни амалга оширади. VMM дискда DOS386.EXE файлида жойлашади.

Windows асосан икки турдаги виртуал машинани ишлатади:

- *Тизимли виртуал машиналар,* уларда Windowsнинг Kernel, User ва GDI компонентлари ҳамда Windows иловалари ишлайди;
- *MS DOS виртуал машиналари,* уларнинг ҳар бирида ҳимояланган ёки 8086 виртуал режимдаги *MS DOS* иловалари (программалари) ишлайди. Windowsда оқим тушунчаси киритилган бўлиб, у тизимли диспетчер томонидан бошқарилувчи махсус объектлардир.

Оқим:

- *процессор ичидаги бажариш муршрутидир;*
- *оқим ишлаётган ихтиёрий 32 разрядли Windows (Win32) иловаси ёки қурилмалар виртуал драйвери томонидан яратилиши мумкин;*
- *хотирани ўзини яратган жараён билан биргаликда ишлатади;*
- *битта жараён томонидан яратилган бир нечта оқимларнинг бири бўлиши мумкин.*

VMM ичида иккита диспетчер ишлайди: асосий диспетчер, у оқимлар приоритетларини (**имтиёзларини**) ҳисоблашга жавоб беради ва квантлаш диспетчери, у оқимларга қанча вақт квантини ажратиш кераклигини аниқлайди.

Диспетчер қуйидаги алгоритм билан ишлайди:

1. *Асосий диспетчер барча оқимларни кўздан кечиради ва уларнинг ҳар бирига бажарилиш приоритеини тайинлайди. Приоритет 0 дан 31 бўлган бутун сон.*
2. *Шундан кейин асосий диспетчер приоритети энг каттасидан кичик бўлган барча оқимларни кутиш ҳолатига ўтказади. Вақт кванти ичида диспетчер кутиш ҳолатидаги оқимларга эътибор бермайди.*
3. *Кейин квант диспетчери ҳар бир оқим бажарилиши учун зарур бўлган вақт квантини ҳисоблайди. Бунда приоритетлар қийматлари ва виртуал машина ҳолати ҳисобига олинади.*
4. Оқимлар ишляпти. Келишилган ҳолда Асосий диспетчер ҳар 20 миллисекунда приоритетларни қата ҳисоблаб чиқади.

Ҳар бир Windows 95 тадбиқий программаси структурлашмаган 4 Гбайт хотирани “кўради”. Бу соҳада унинг ўзи, тизимли код (программалар) ва Windows 95 драйверлари жойлашади. 32 разрядли программа шахсий компбютердан битта ўзи монопол равишда фойдаланаётгандек, яъни 32 разрядли программалар бир-бирини “кўрмайди”. Бироқ улар алмашиш буферлари (Cliboard), DDE ва OLE механизмлари орқали берилганларни алмашишлари мумкин. Барча 32 разрядли тадбиқий программалар алоҳида оқимларни бошқаришга асосланган сиқиб чиқарувчи кўпмасалалик моделига мос равишда бажарилади. Сиқиб чиқарувчи режалаштириш мультипрограммалик механизмни нисбатан равон ва ишончли амалга оширишга имкон беради.

4 Гбайт	Компоненталар
3 Гбайт	Win16 тадбиқий программалари. Тизимли DLL биргаликда ишлатиладиган соҳа.
2 Гбайт	Тадбиқий программалар
4 Мбайт	Реал режим компоненталари
64 Кбайт	
0 байт	

Windows хотира картаси

Юқоридаги расмда келтирилган Windows хотира картаси қуйидаги ҳолда ишлатилади. Хотиранинг 0 ва 4 Мб оралиғини барча жараёнлар биргаликда ишлатади. Бу реал режимдаги қурилма драйверлари ва резидент программалар ва айрим 16 разрядли Windows программаларини ўзаро келишиб ишлашига имкон беради. 32 разрядли программалар бошланғич 64 Кбайт хотирага мурожат қила олмайди. 4 Мбйт ва 2 Гбайт оралиғида ҳар бир Win32 программалари учун структуралашмаган адрес фазоси ҳисобланади; у бошқа Win32 жараёнларини кўрмайди. 2 ва 3 Гбайт адресли хотирада тизимли Windows DLL (Kernel,GDI, User) ва Win16 программалари жойлашади. 16 разрядли Windows программалари мультипрограммалик иш режимда адрес фазосидан биргаликда фойдаланган ҳолда ишлайди. 3 ва 4Гбайт адрес оралиғида операцион тизмнинг 0-ҳалқа компоненталари жойлашади: виртуал машина ва файлларни бошқариш тизимлари. Бу соҳага барча Win32 программалари мурожаат қилиши мумкин. Унда ҳар бир Win32 программаси учун алоҳида адрес фазоси ва умумий хотираси жойлашади.

Назорат саволлари

1. Windows ОТда қандай қилиб бир вақтда бир нечта программа ишлайди?
2. Ҳимояланган иш режими деганда нима тушунилади?
3. Windows ОТ 4 Гбайтли хотирадан қандай фойдаланади?
4. Windows ОТ ишлашида. Виртуал машина ролини тушунтириб беринг.