8-дәріс. Жадты басқару

8.1. Жадты басқаруға қойылатын талаптар
Ауыстыру
Қорғау
Ортақ пайдалану
Логикалық ұйым
Физикалық ұйым
8.2. Жадты үлестіру
Бекітілген үлестіру
Бөлім өлшемдері
Орналастыру алгоритмі
Динамикалық үлестіру
Орналастыру алгоритмі
Алмастыру алгоритмі
Ауыстыру
8.3. Жадты ұйымдастыру
8.4. Сегменттеу

Көп функциялы жүйелерде негізгі жад екіге бөлінеді: бір бөлігі операциялық жүйе үшін (резиденттік монитор, ядро), ал екінші "пайдаланушы" жад бөлігі бірнеше процестерді орналастыру үшін. Бұл бөлу тапсырмасын операциялық жүйе динамикалық түрде орындайды және жадты деп аталады. Жадты тиімді бөлу, оған мүмкіндігінше көп процестерді орналастыруға мүмкіндік береді.

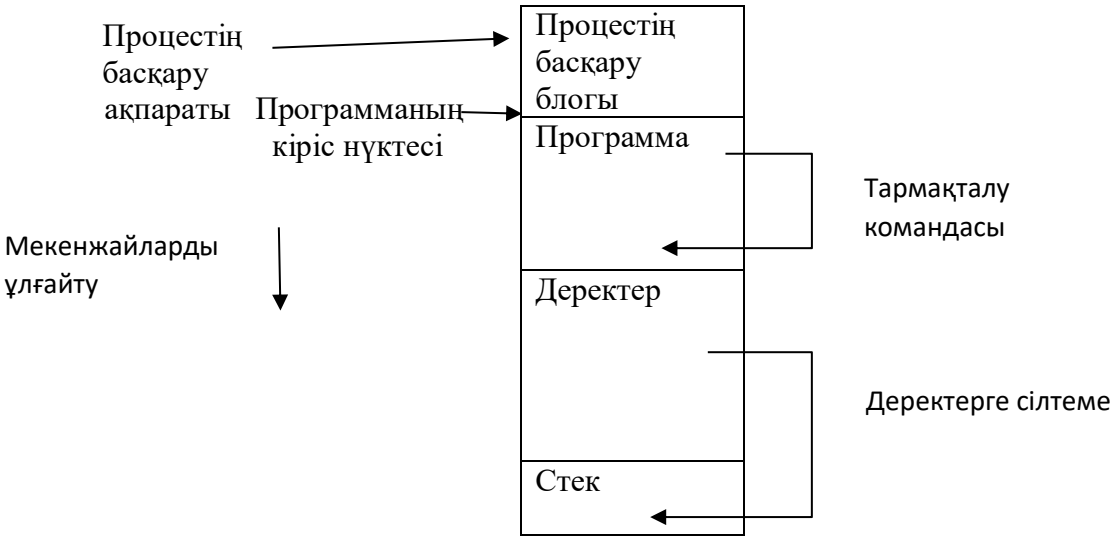
8.1. Жадты басқаруға қойылатын талаптар

Жадты басқаруға байланысты әртүрлі механизмдер мен стратегияларды қарастырған кезде, олар қанағаттандыруы керек талаптарды есте ұстаған жөн. Бұл талаптар мыналарды қамтиды.

- Ауыстыру.
- Қорғаныс.
- Ортақ пайдалану.
- Логикалық ұйым.
- Физикалық ұйым.

Ауыстыру

Көп функциялы жүйеде қол жетімді негізгі жад жалпы жағдайда көптеген процестер арасында бөлінеді. Әдетте бағдарламаның қай жерде орналасатыны алдын-ала белгісіз, сонымен қатар свопинг кезінде бағдарламаны жадтың бір аймағынан екіншісіне ауыстыруға болады. Бұл жағдайлар суретте көрсетілген адресацияға белгілі бір техникалық талаптардың болуын анықтайды. 8.1-суретте процестің бейнесі көрсетілген. Қарапайым болу үшін процестің кескіні негізгі жадтың бір үздіксіз аймағын алады делік. Операциялық жүйе процестің басқару ақпаратын және орындалу стекінің орналасқан жерін, сондай-ақ процесті бастау үшін кіру нүктелерін білуі керек. Операциялық жүйе жадты басқарумен айналысатындықтан және ол процесті негізгі жадқа орналастыратындықтан, ол автоматты түрде тиісті мекен-жайларды алады. Алайда, операциялық жүйе көрсетілген ақпаратты алумен қатар, процесс бағдарламаның өзінде жадқа қол жеткізе алуы керек. Сонымен, тармақтау командаларында олардан кейін орындалуы керек командаларды көрсететін мекен-жайлар бар; деректерге жүгіну командалары-олар жұмыс істейтін байттардың немесе сөздердің мекен-жайы. Қалай болғанда да, процессор мен амалдық жүйенің бағдарламалық жасақтамасы бағдарлама кодындағы сілтемелерді негізгі жадтағы бағдарламаның ағымдағы орналасуына сәйкес келетін нақты физикалық мекенжайларға аудара алуы керек.



8.1-сурет. Процесті адрестеуге қойылатын талаптар

Қорғаныс

Әрбір процесс кездейсоқ немесе қасақана басқа процестердің жағымсыз әсерінен қорғалуы керек. Сондықтан, басқа процестердің коды оқу немесе жазу үшін берілген процестің жадына рұқсатсыз қол жеткізе алмауы керек. Алайда, қозғалыс сұранысын қанағаттандыру қорғаныс міндетін қиындатады. Бұл ретте, жадты қорғау талаптары операциялық жүйе (бағдарламалық қамтамасыз ету) деңгейінде емес, процессор (аппараттық қамтамасыз ету) деңгейінде қанағаттандырылуы тиіс, себебі операциялық жүйе бағдарлама орындайтын жадқа барлық жүгінулерді алдын ала болжай алмайды. Тиісті аппараттық мүмкіндіктер-бұл бағдарлама жұмыс істеп тұрған кезде жадқа (деректерге немесе кодқа) қол жетімділікті анықтайтын жалғыз құрал.

Ортақ пайдалану

Кез-келген қорғаныс механизмі бірнеше процестерге негізгі жадтың бір аймағына қол жеткізуге мүмкіндік беру үшін жеткілікті икемділікке ие болуы керек. Белгілі бір тапсырмада жұмыс істейтін процестер бірдей деректер құрылымдарына бірлесіп қол жеткізуді қажет етуі мүмкін. Жадты басқару жүйесі осылайша жадтың бөлінген аймақтарына басқарылатын қол жетімділікті қамтамасыз етуі керек, ал жад қорғанысын әлсіретпейді.

Логикалық ұйым

Әрдайым дерлік, компьютерлік жүйеде негізгі жад байттар немесе сөздер тізбегінен тұратын сызықтық (бір өлшемді) адрестік кеңістік ретінде ұйымдастырылған. Екінші реттік жад өзінің физикалық деңгейінде де ұйымдастырылған. Көптеген бағдарламалар модульдер түрінде ұйымдастырылған, олардың кейбіреулері өзгермейді (тек оқу үшін, тек орындау үшін), ал басқаларында өзгертуге болатын мәліметтер бар. Пайдаланушы бағдарламаларымен және модульдермен ұсынылған мәліметтермен жұмыс істеу бірқатар артықшылықтарды қамтамасыз етеді.

Модульдерді бір-бірінен тәуелсіз құруға және құрастыруға болады, ал бір модульден екінші модульге барлық сілтемелер бағдарлама жұмыс істеп тұрған кезде жүйемен шешіледі.

1. Әр түрлі модульдер өте қарапайым үстеме шығындарға байланысты әр түрлі қорғаныс дәрежесін ала алады (тек оқу үшін, тек орындау үшін).
2. Модульдерді әртүрлі процестермен бөлісуді қамтамасыз ететін механизмді қолдануға болады. Модуль деңгейінде бөлісуді қамтамасыз етудің басты артықшылығы-

олар бағдарламашының тапсырмаға деген көзқарасына сәйкес келеді, сондықтан белгілі бір модульді бөлісу қажет пе, жоқ па, соны анықтау оңайырақ.

Осы талаптарды жақсы қанағаттандыратын құрал-бұл сегментация.

Физикалық ұйым

Компьютердің жады кем дегенде екі деңгейге бөлінеді: негізгі және екінші деңгей. Негізгі жад салыстырмалы түрде жоғары бағамен жылдам қол жетімділікті қамтамасыз етеді; сонымен қатар, ол өзгермелі, яғни ұзақ мерзімді сақтауды қамтамасыз етпейді. Екінші жад баяу және негізгі жадқа қарағанда арзан және әдетте өзгермейтін болады. Сондықтан, екінші сыйымдылықты жад бағдарламалар мен деректерді ұзақ уақыт сақтауға қызмет ете алады, ал кіші сыйымдылықтың негізгі жады қазіргі уақытта қолданылатын бағдарламалар мен деректерді сақтауға қызмет ете алады. Мұндай екі деңгейлі құрылымда негізгі және екінші жад арасындағы ақпарат ағынын ұйымдастыру операциялық жүйеге жүктеледі.

8.2. Жадты үлестіру

Жадты басқарудың негізгі жұмысы-оны процессор орындау үшін бағдарламаны негізгі жадқа орналастыру. Қазіргі заманғы көп функциялы жүйелерде бұл тапсырма виртуалды жад деп аталатын күрделі сызбаны қолдануды қамтиды. Виртуалды жад, өз кезегінде, бір немесе екі негізгі технологияны - сегменттеу (сегментация) және бетті жадты ұйымдастыруға (paging) негізделген. Виртуалды жадты ұйымдастырудың осы әдістерін қарастырмас бұрын, біз виртуалды жадпен байланысты емес қарапайым әдістермен танысуымыз керек (8.1-кесте).

8.1-кесте. Жадты басқару технологиялары

Технология	Сипаттама	Күшті жақтары	Әлсіз жақтары
Бекітілген үлестіру	Негізгі жад жүйені құру кезінде бірқатар статикалық бөлімдерге бөлінеді. Процесті тең немесе үлкен бөлімге жүктеуге болады	Іске асыру оңай, шағын жүйелік үстеме	Ішкі фрагментацияға байланысты жадты тиімсіз пайдалану, Белсенді процестердің белгіленген саны
Динамикалық үлестіру	Бөлімдер динамикалық түрде жасалады; әр процесс қатаң қажетті бөлімнің бөліміне жүктеледі	Ішкі фрагментация жоқ, негізгі жадты тиімді пайдалану	Сыртқы фрагментацияға қарсы тұру үшін тығыздау қажеттілігіне байланысты процессорды тиімсіз пайдалану
Қарапайым бетті ұйымдастыру	Негізгі жад тең мөлшерде бірқатар кадрларға бөлінеді. Әр процесс бірдей өлшемдегі және кадрмен бірдей ұзындықтағы белгілі бір беттерге бөлінеді. Процесс	Сыртқы фрагментация жоқ	Шағын ішкі фрагментацияның болуы

	оның барлық беттерін қол жетімді, бірақ міндетті түрде дәйекті кадрларға жүктеу арқылы жүктеледі		
Жай сегменттеу	Әр процесс бірқатар сегменттерге бөлінеді. Процесс барлық сегменттерін динамикалық (міндетті емес) бөлімдерге жүктеу арқылы жүктеледі	Ішкі фрагментация жоқ; динамикалық таратумен салыстырғанда жадыны пайдалану тиімділігінің жоғарылауы және үстеме шығындардың төмендеуі	Сыртқы фрагментация
Виртуалды жадты беттік ұйымдастыру	Барлығы, қарапайым Парақ ұйымындағыдай, процестің барлық беттерін бір уақытта жүктеудің қажеті жоқ. Қажетті бейрезиденттік беттер автоматты түрде жадқа жүктеледі	Сыртқы фрагментация жоқ; көпмүшеліктің жоғары дәрежесі; үлкен виртуалды мекен-жай кеңістігі	Жадыны басқару жүйесінің күрделілігіне байланысты үстеме шығындар
Виртуалды жадты сегментациялау	Барлығы, қарапайым сегментация сияқты, ерекшелік бар. Процестің барлық сегменттерін бір уақытта жүктеудің қажеті жоқ. Қажетті резидент емес сегменттер автоматты түрде жадқа жүктеледі	Ішкі фрагментация жоқ; көп деңгейлі көп деңгейлі; үлкен виртуалды мекен-жай кеңістігі; қорғауды және бөлісуді қолдау	Жадыны басқару жүйесінің күрделілігіне байланысты үстеме шығындар

Бекітілген үлестіру

Бұл қол жетімді жадты басқарудың қарапайым сызбасы – оны белгіленген шекаралары бар аймақтарға бөлу. 8.2-суретте бекітілген үлестірімнің екі нұсқасы

көрсетілген. Бір мүмкіндік-бірдей мөлшердегі бөлімдерді пайдалану. Бұл жағдайда өлшемі бөлімнің өлшемінен аспайтын кез-келген процесті кез-келген қол жетімді бөлімге жүктеуге болады. Егер барлық бөлімдер бос болмаса және дайын немесе жұмыс күйінде бірде-бір процесс болмаса, амалдық жүйе процесті кез-келген бөлімнен шығарып, басқа процесті жүктей алады, осылайша процессорды жұмыс істеуге мүмкіндік береді.

Операциялық Жүйе 8М
8М
8М
8М
8М
8М
8М
8М

Операциялық Жүйе 8М
2М
4М
6М
8М
8М
12М
16М

а) бірдей мөлшердегі бөлімдер б) әртүрлі мөлшердегі бөлімдер

8.2-сурет. 64 мегабайт жадының тұрақты үлестірілуінің мысалы

Бірдей өлшемдегі бөлімдерді пайдалану кезінде екі қиындық бар.

* Бағдарлама бөлімде орналастыру үшін тым үлкен болуы мүмкін.

* Негізгі жадты пайдалану өте тиімсіз. Кез-келген бағдарлама, оның көлеміне қарамастан, бүкіл бөлімді алады. Пайдаланылмаған жадтың пайда болуының бұл құбылысы, жүктелетін блоктың өлшемі бөлімнен кіші болғандықтан, ішкі фрагментация деп аталады (ішкі фрагментация).

Бұл қиындықтармен әртүрлі мөлшердегі бөлімдерді қолдану арқылы күресуге болады (8.2-сурет).

Орналастыру алгоритмі

Бөлімдер бірдей мөлшерде болған жағдайда, процестерді жадқа орналастыру тривиальды міндет болып табылады. Еркін бөлімдердің қайсысы процестің орналастырылатыны маңызды емес. Егер барлық бөлімдер тез арада жұмыс істеуге дайын емес процестермен айналысса, олардың кез-келгенін жаңа процесс үшін жадты босату үшін жүктеуге болады.

Бөлімдер әртүрлі мөлшерде болған кезде, процестерді жад бөлімдеріне тағайындаудың екі мүмкін әдісі бар. Қарапайым жол-бұл әр процесті осы процесті толығымен орналастыра алатын ең кіші бөлімде орналастыру. Бұл тәсілдің артықшылығы-процестерді ішкі фрагментацияны азайту үшін жад бөлімдері арасында бөлуге болады. Бірдей мөлшердегі бөлімдерді қолданумен салыстырғанда әртүрлі мөлшердегі бөлімдерді қолдану бұл әдіске қосымша икемділік береді. Сонымен қатар, бекітілген бөлімдері бар схемалар салыстырмалы түрде қарапайым, операциялық

жүйеге минималды талаптар қояды; процессордың үстеме шығындары аз. Алайда, бұл схемалардың елеулі кемшіліктері бар.

Жүйені құру кезінде анықталған бөлімдер саны белсенді (тоқтатылмаған) процестердің санын шектейді.

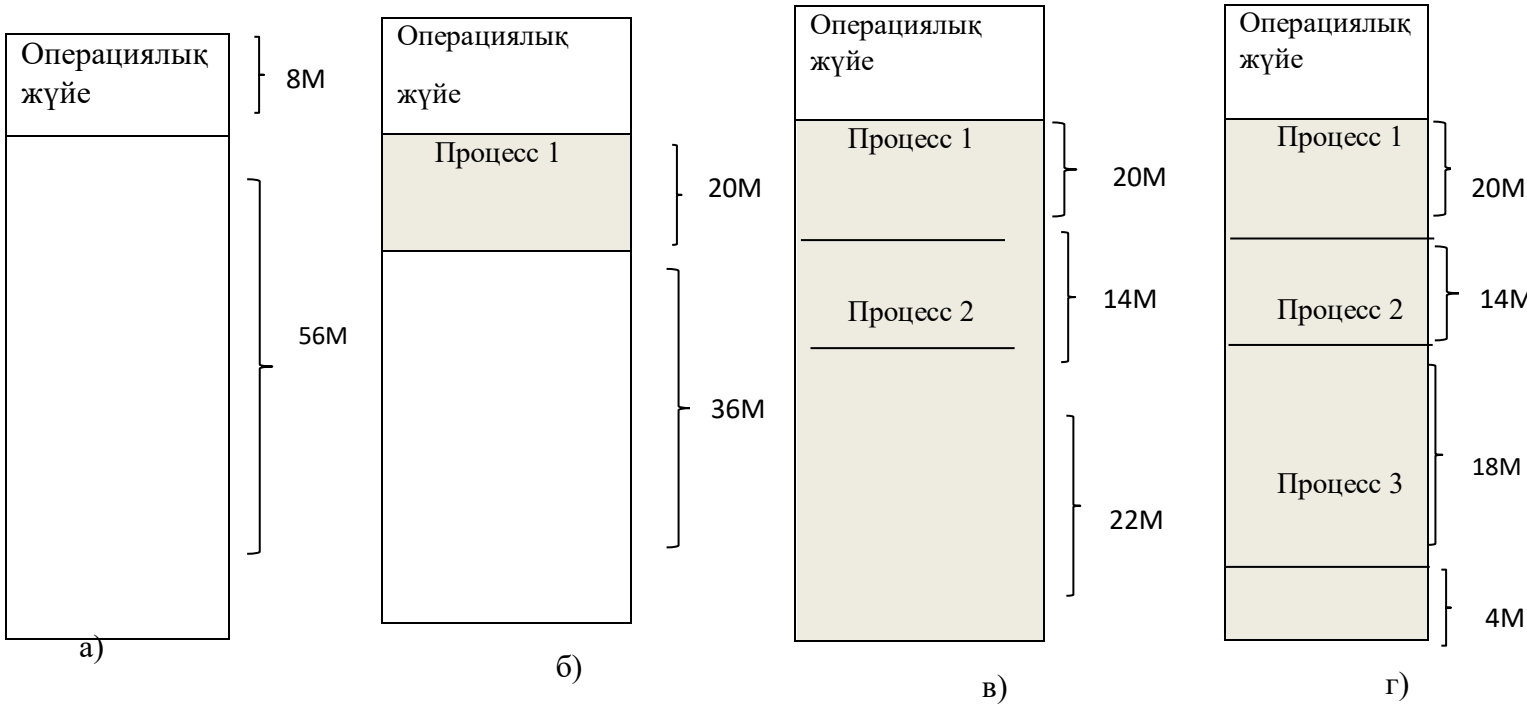
Бөлімдер мөлшері алдын-ала орнатылғандықтан, жүйені құру кезінде кішігірім процестер жадтың тиімсіз пайдаланылуына әкеледі. Қазіргі уақытта тұрақты бөлу іс жүзінде қолданылмайды.

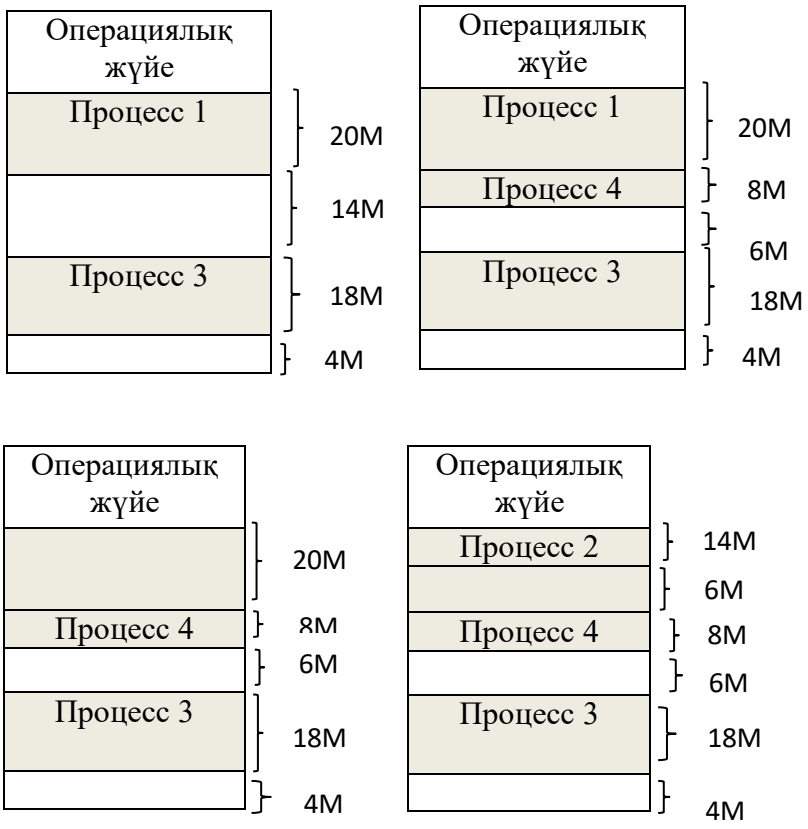
Динамикалық үлестіру

Тұрақты үлестірумен байланысты қиындықтарды жеңу үшін динамикалық үлестіру деп аталатын балама тәсіл жасалды. Қазіргі уақытта бұл тәсіл жадыны басқарудың неғұрлым күрделі және тиімді технологияларымен алмастырылуда.

Динамикалық бөлу кезінде айнымалы ұзындықтың ауыспалы бөлімдерінің саны қалыптасады. Процесті негізгі жадқа орналастырған кезде, ол үшін жадтың қатаң қажетті мөлшері бөлінеді, одан басқа ештеңе жоқ. Мысал ретінде 64 Мб негізгі жадты қолдануды қарастырыңыз (8.3-сурет). Бастапқыда операциялық жүйе қолданатын аймақты қоспағанда, барлық жад бос (8.3, а сурет). Алғашқы үш процесс жадқа операциялық жүйе аяқталатын мекен-жайдан бастап және осы процесс қажет болғанша жадты қолдана отырып жүктеледі (8.3-сурет, B-G). Осыдан кейін, негізгі жадтың соңында төртінші процесті орналастыру үшін тым кішкентай "тесік" қалады. Бір сәтте жадтағы барлық процестер белсенді емес болып шығады және операциялық жүйе екінші процесті (7.4-сурет, д) жүктейді, содан кейін жаңа, төртінші процесті (7.4-сурет, е) жүктеу үшін жеткілікті жад қалады. 4-процесс 2-ші процесстен кіші болғандықтан, жадта тағы бір кішкентай "тесік" пайда болады. Белгілі бір уақытта жадтағы барлық процестер белсенді емес, бірақ 2-процесс жұмыс істеуге дайын болғаннан кейін, ол үшін жадта бос орын жоқ, ал операциялық жүйе қажетті орынды босату үшін 1-процесті түсіруге мәжбүр болады (7.4-сурет, ж) және 2-процесті негізгі жадқа орналастырыңыз (7.4-сурет, з).

8.3-сурет. Динамикалық үлестіру мысалы





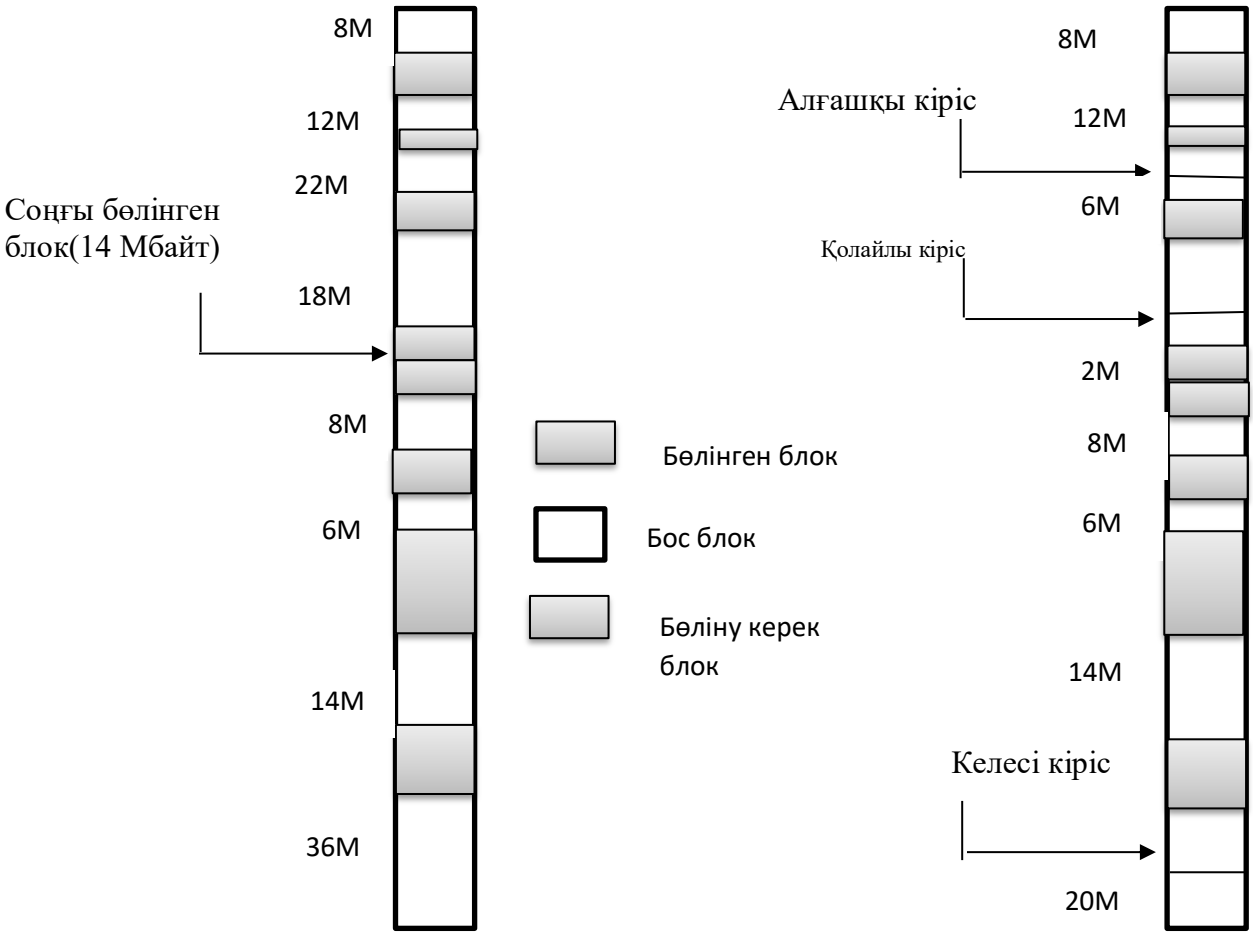
Уақыт өте келе, жад көп бөлінеді және оны пайдалану тиімділігі төмендейді. Бұл құбылыс сыртқы фрагментация деп аталады.

Бұл құбылысты жеңудің бір әдісі-тығыздау (compaction): анда-санда Операциялық жүйе жадтағы процестерді жадтың іргелес аймақтарын алатындай етіп жылжытады; бос жад бір блокқа жиналады. Тығыздағышты қолданудың күрделілігі-бұл қосымша уақытты қажет етеді.

Орналастыру алгоритмі

Жадтың тығыздалуы процессордың қосымша уақытын қажет ететіндіктен, операциялық жүйені жасаушы процестерді жадқа қалай қою керектігі туралы ақылға қонымды шешім қабылдауы керек (бейнелі түрде, тесіктерді қалай жабу керек). Процесті негізгі жадқа жүктеу сәті келгенде және жеткілікті мөлшерде бірнеше бос жад блоктары болған кезде, амалдық жүйе қай бос блокты пайдалану керектігін шешуі керек.

Үш негізгі алгоритмді қарастыруға болады - ең жақсы қолайлы, бірінші қолайлы, келесі қолайлы. Олардың барлығы процесті орналастыру үшін жеткілікті мөлшерде еркін блоктарды таңдаумен шектеледі. Ең жақсы қолайлы әдіс өлшемі талап етілгенге жақын блокты таңдайды; бірінші қолайлы әдіс жадтың басынан бастап барлық бос блоктарды тексереді және процесті орналастыру үшін бірінші өлшемді таңдайды. Келесі қолайлы әдіс бірінші қолайлы әдіс сияқты жұмыс істейді, бірақ ол тексеруді блок соңғы рет бөлінген жерден бастайды және процестерді жадтан орналастырғаннан және түсіргеннен кейін жад конфигурациясының мысалын көрсетеді (8.4-суретті қараңыз).



а)Бөлінгенге дейін б)Бөлінгеннен кейін

8.4-сурет. 16 МБ блокты бөлгенге дейін және кейін жад конфигурациясының мысалы

Алмастыру алгоритмі

Динамикалық үлестірімді қолдана отырып, көп функциялы жүйеде негізгі жадтағы барлық процестер Құлыпталған күйде болатын сәт пайда болады, ал тығыздаудан кейін де қосымша процесс үшін жад жеткіліксіз. Белсенді процестің босатылуын күту үшін процессордың уақытын жоғалтпау үшін амалдық жүйе процестердің бірін негізгі жадтан шығарып, жаңа процесске немесе процеске дайын күйде орын бере алады. Операциялық жүйенің міндеті-қандай процесті жадтан шығару керектігін анықтау. Ауыстыру алгоритмінің тақырыбы виртуалды жадтың әртүрлі схемаларына байланысты егжей-тегжейлі қарастырылатындықтан, біз осы мәселені талқылауды кейінге қалдырамыз.

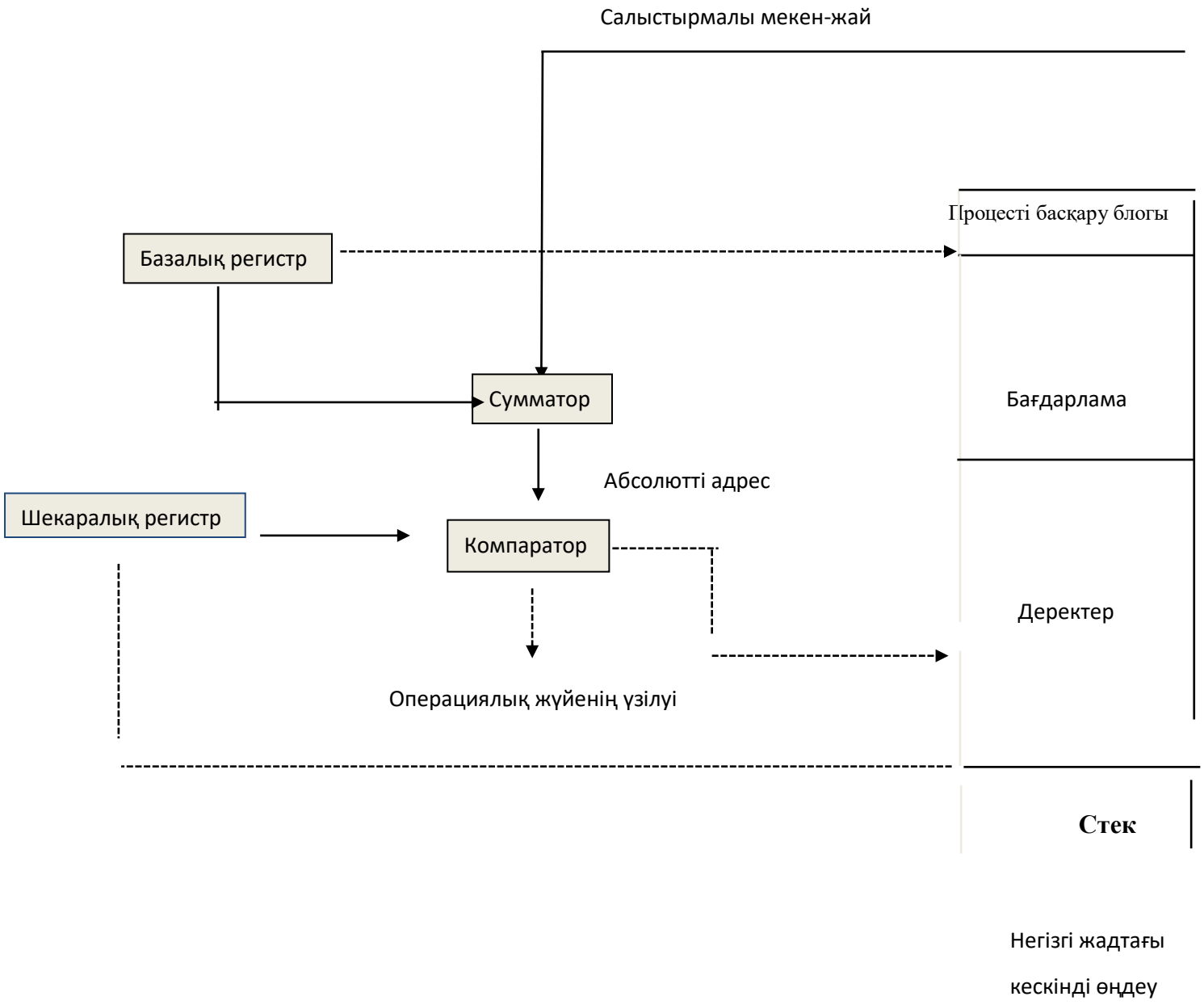
Ауыстыру

Процесс қолданылатын командалар мен деректердің орналасуы тұрақты емес және процесті жүктеу және жүктеу (немесе жылжыту) кезінде әр уақытта өзгереді. Бұл мәселені шешу үшін мекен-жай түрлерін ажырату керек. Логикалық адрес-бұл жадтағы деректердің ағымдағы орналасуына тәуелсіз жад ұяшығына сілтеме; осы жад ұяшығына кірмес бұрын логикалық адресті физикалық мекен-жайға жіберу керек. Салыстырмалы мекен - жай дегеніміз-белгілі бір нүктеге (әдетте бағдарламаның басталуына) қатысты позициямен анықталатын логикалық мекен-жайдың ерекше жағдайы. Физикалық мекен-жай (абсолютті деп те аталады) - бұл бізді қызықтыратын негізгі жад ұяшығының нақты орналасуы.

Жадта салыстырмалы мекен-жайларды қолданатын бағдарлама жұмыс уақытын динамикалық жүктеу арқылы жүктеледі. Әдетте, жүктелген процестегі барлық жад сілтемелері осы бағдарламаның басталуына қатысты берілген. Осылайша,

бағдарламаның дұрыс жұмыс істеуі үшін жадқа кіретін команданы орындау кезінде салыстырмалы мекен-жайларды физикалық мекен-жайларға жіберетін аппараттық механизм қажет.

8.5-суретте мекенжайды аудару әдісі көрсетілген. Процесс орындалу күйіне өткен кезде, процессордың арнайы регистріне, кейде базалық деп аталады, процестің бастапқы мекен-жайы Негізгі жадқа жүктеледі. Сонымен қатар, бағдарламаның соңғы жад ұяшығының мекен-жайы бар" шекара " (bounds) регистрі қолданылады. Бұл мәндер бағдарламаны негізгі жадқа жүктеген кезде регистрлерге енгізіледі.



8.5-сурет. Аппараттық қозғалысты қолдау

Процесті орындау кезінде командаларда кездесетін салыстырмалы мекен-жайларды процессор екі кезеңде өңдейді. Біріншіден, абсолютті мекенжайды алу үшін салыстырмалы мекен-жайға негізгі регистрдің мәні қосылады. Содан кейін алынған абсолютті адрес шекаралық регистрдегі мәнмен салыстырылады. Егер алынған

абсолютті мекен-жай осы процеске тиесілі болса, команда орындалуы мүмкін; әйтпесе, осы қатеге сәйкес келетін операциялық жүйенің үзілуі жасалады.

8.5-суретте көрсетілген Схема оларды орындау барысында бағдарламаларды негізгі жадқа түсіру және жүктеу мүмкіндігін қамтамасыз етеді; сонымен қатар, әр процестің бейнесі негізгі және шекаралық регистрлердегі мекен-жайлармен шектеледі және осылайша басқа процестердің қажетсіз қол жетімділігінен қорғалған.

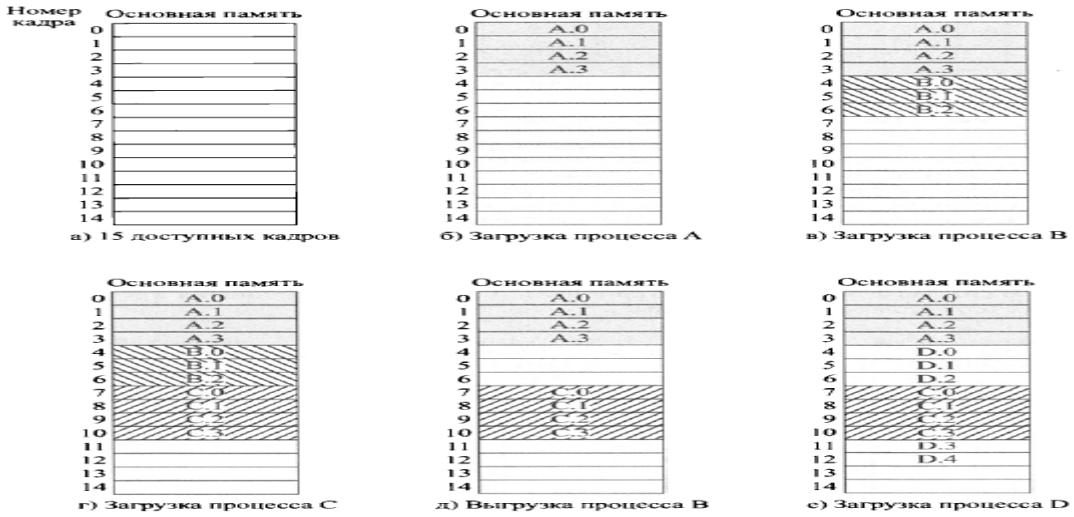
8.3. Бетті есте сақтауды ұйымдастыру

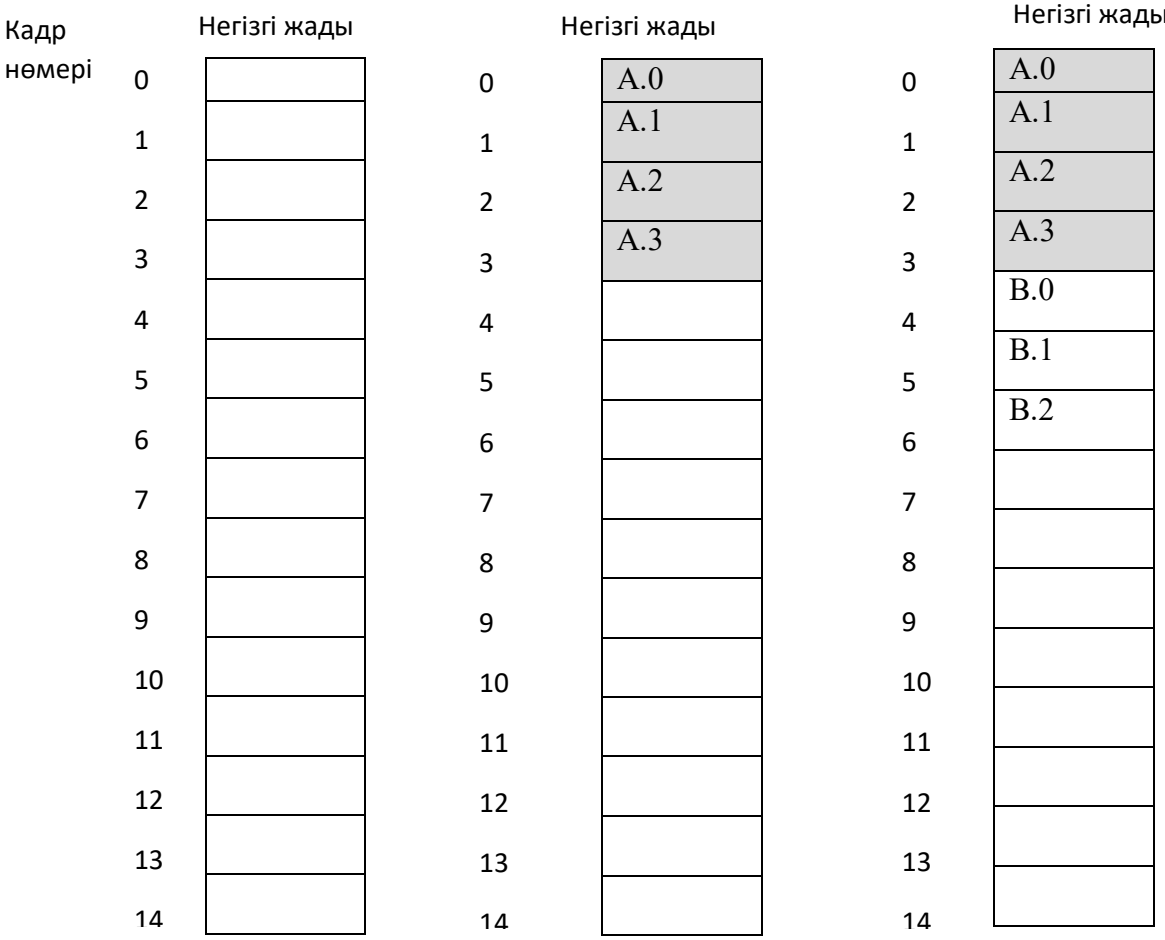
Әр түрлі өлшемдері бар бөлімдер де, ауыспалы бөлімдер де жадты тиімді пайдаланбайды. Біріншісінің нәтижесі – ішкі фрагментация, соңғысының нәтижесі-сыртқы. Алайда, негізгі жад салыстырмалы түрде кішкентай, бекітілген өлшемдегі бірдей блоктарға бөлінген делік. Содан кейін беттер (Беттер) деп аталатын процесс блоктары кадрлар (жақтаулар) немесе жақтаулар деп аталатын бос жад блоктарымен байланысты болуы мүмкін. Әр кадрда бір деректер парағы болуы мүмкін. Осы бөлімнен білетін жадты ұйымдастыру кезінде сыртқы фрагментация мүлдем болмайды, ал ішкі фрагментацияға байланысты шығындар процестің соңғы бетінің бір бөлігімен шектеледі.

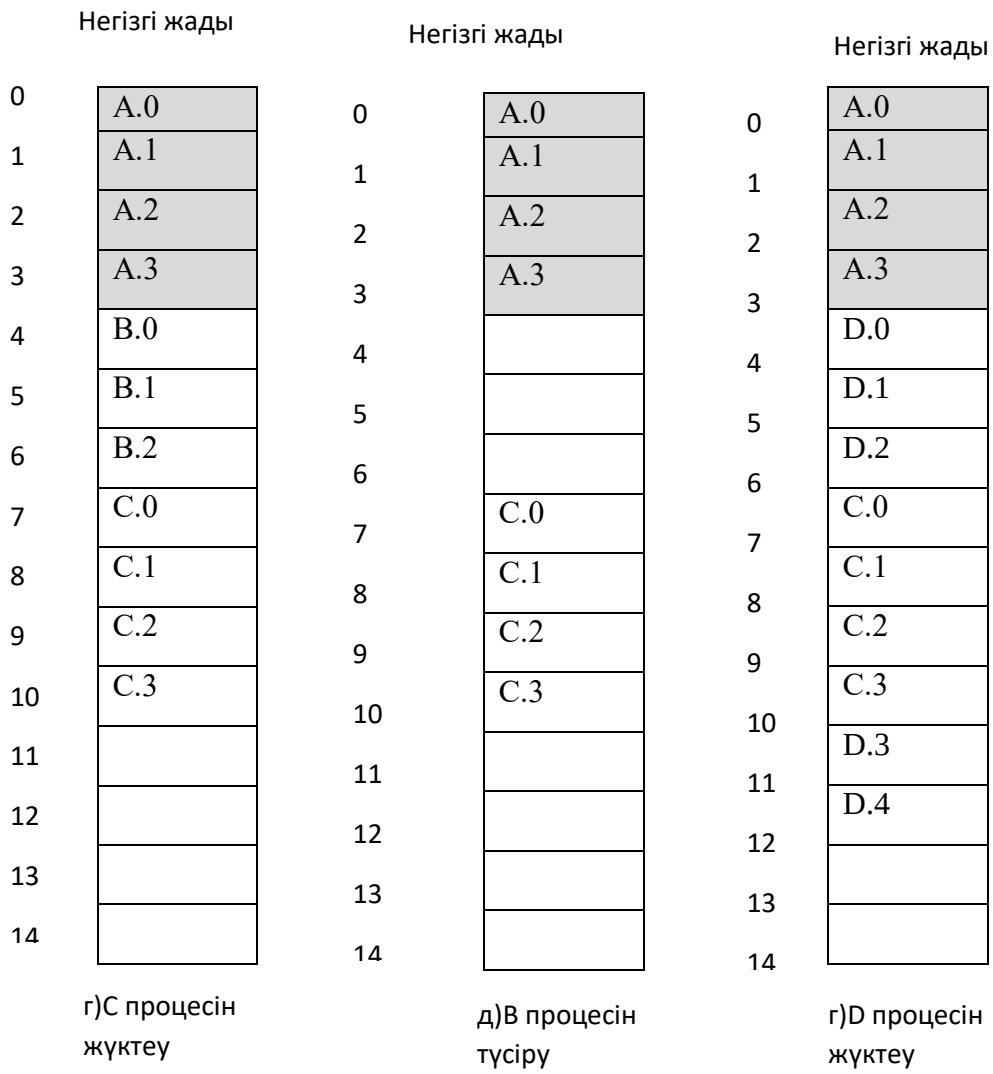
8.6-суретте беттер мен жақтауларды пайдалану көрсетілген. Кез-келген уақытта кейбір жад кадрлары қолданылады, ал кейбіреулері бос. Операциялық жүйе бос кадрлардың тізімін қолдайды. Дискіде сақталған a процесі төрт беттен тұрады. Бұл процесті жадқа жүктеу уақыты келгенде, амалдық жүйе төрт бос жақтауды тауып, a процесінің беттерін осы жақтауларға жүктейді (8.6-сурет, B). Содан кейін үш беттен тұратын b процесі және төрт беттен тұратын c процесі жүктеледі. Осыдан кейін b процесі тоқтатылады және негізгі жадтан шығарылады. Кейінірек жадтағы барлық процестер құлыпталып, Операциялық жүйе бес беттен тұратын жаңа d процесін жадқа жүктейтін сәт келеді. Алайда, бүкіл процесті орналастыру үшін жеткілікті кадрлардың бір үздіксіз аймағы болмаған жағдайда бір негізгі мекен-жай регистрі жеткіліксіз және әр процесс үшін операциялық жүйе парақ кестесін қолдауы керек. Бет кестесі процестің әр бетінің кадрларының орналасқан жерін көрсетеді. Бағдарлама ішінде логикалық мекен-жай бет нөмірінен және оның ішіндегі офсеттен тұрады. Бетті ұйымдастыру кезінде логикалық адрестерді физикалық адрестерге түрлендіру процессор шешетін аппараттық деңгейдің міндеті болып қала береді.

Енді процессорда ағымдағы процестің бет кестесі қайда екендігі туралы ақпарат болуы керек. Ұсынылған логикалық мекен-жай (бет нөмірі және офсеттік) процессор бет кестесін физикалық мекен-жайға (кадр нөмірі, офсеттік) айналдырады.

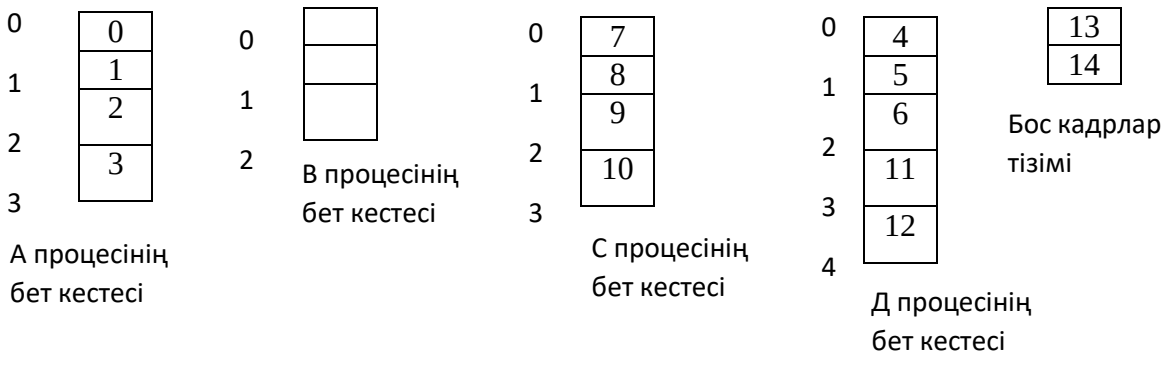
8.7-суретте беттердің әртүрлі кестелері көрсетілген. Бет кестесінде процестің әр беті үшін бір жазба бар, сондықтан кестені 0-ден бастап парақ нөмірімен индекстеу оңай. Әр жазбада тиісті бет сақталатын негізгі жадтағы кадр нөмірі бар (егер бар болса). Сонымен қатар, операциялық жүйе бос кадрлардың бірыңғай тізімін қолдайды (яғни, ешқандай процеспен айналыспайды және оларға беттерді орналастыруға болады).


Номер кадра
Основная память
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14
а) 15 доступных кадров
A.0 A.1 A.2 A.3
б) Загрузка процесса A
A.0 A.1 A.2 A.3 B.0 B.1 B.2
в) Загрузка процесса B
A.0 A.1 A.2 A.3 B.0 B.1 B.2 C.0 C.1 C.2 C.3
г) Загрузка процесса C
A.0 A.1 A.2 A.3 C.0 C.1 C.2 C.3
д) Выгрузка процесса B
A.0 A.1 A.2 A.3 D.0 D.1 D.2 C.0 C.1 C.2 C.3 D.3 D.4
е) Загрузка процесса D


Кадр нөмері
Негізгі жады
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14
а)Қолжетімді 15 кадр
A.0 A.1 A.2 A.3
б)А процесін жүктеу
A.0 A.1 A.2 A.3 B.0 B.1 B.2
б)В процесін жүктеу



8.6-сурет. Беттерді бос кадрлар бойынша бөлу



8.7-сурет. 8.6(е)-суреттегі үлгіге сәйкес келетін деректер құрылымы

Осылайша, мұнда сипатталған қарапайым бетті ұйымдастыру тұрақты таратуға ұқсас. Айырмашылықтар бөлімдердің өте аз мөлшерінде, олар сонымен қатар іргелес болмауы мүмкін. Осындай схемамен жұмыс істеудің ыңғайлылығы үшін біз парақтың өлшемі (демек, кадр өлшемі) 2 дәрежесі болуы керек ережені қосамыз.

8.4. СЕГМЕНТТЕУ

Пайдаланушы бағдарламасын бөлудің балама әдісі-сегментация. Бұл жағдайда бағдарлама және онымен байланысты деректер бірқатар сегменттерге бөлінеді. Сегменттің максималды мөлшері болса да, тең өлшем шарты сегменттерге қолданылмайды. Бетті ұйымдастырудағы сияқты, логикалық мекен-жай екі бөліктен тұрады, бұл жағдайда – сегмент нөмірі және орын ауыстыру.

Әр түрлі мөлшердегі сегменттерді қолдану бұл әдісті жадтың динамикалық таралуына ұқсас етеді. Егер қабаттасулар мен виртуалды жад пайдаланылмаса, онда бағдарламаны орындау үшін оның барлық сегменттері жадқа жүктелуі керек; алайда, динамикалық үлестіруден айырмашылығы, бұл жағдайда сегменттер бірнеше бөлімдерді алуы мүмкін, олар сонымен қатар іргелес болмауы мүмкін.

Сегменттердің әртүрлі мөлшерде болуының тағы бір салдары-логикалық және физикалық адрестер арасында қарапайым байланыс болмауы. Бетті ұйымдастыруға ұқсас, қарапайым сегментация схемасы әр процесс үшін сегменттер кестесін және негізгі жадтың бос блоктарының тізімін пайдаланады. Сегменттер кестесінің әр жазбасында жүйені дұрыс емес мекенжайларды пайдаланудан қорғау үшін негізгі жадтағы сегменттің бастапқы мекен-жайы және оның ұзындығы болуы керек. Процесс жұмыс істеген кезде оның сегменттер кестесінің мекен-жайы жадыны басқарудың аппараттық құралы пайдаланатын арнайы тізілімге енгізіледі.

ҚОЛДАНЫЛҒАН ӘДЕБИЕТТЕР ТІЗІМІ

1. Garg, R.; Verma, G.Operating Systems [OP]: An Introduction - Softcover Publisher: Mercury Learning & Information, 2017. 290 p.
2. https://gifer.com/ru/7h0m
3. https://3dnews.ru/1034959
4. Darrell Hajek, Cesar Herrera, Flor Narciso Principles of Operating Systems. Independently Published (24 April 2020) 176 pages.
5. Andrew S. Tanenbaum and Herbert Bos. Modern Operating Systems. 4/E.1136 pages, Pearson India, 2016.
6. Silberschatz Abraham, Galvin Peter Baer and Gadne Greg. Operating system concepts.
7. Amdahl GM (1967) Validity of the single-processor approach to achieve large scale computing capabilities. AFIPS Joint Spring Conference Proceedings 30 (Atlantic City, NJ, Apr. 18–20), AFIPS Press, Reston VA, pp 483–485.
8. https://studfile.net/.
9. https://habr.com/ru/post/40227/.
10. wikimedia.org
11. wordpress.com
12. blackandwhitecomputer.blog
13. http://www-inst.eecs.berkeley.edu/~n252/paper/Amdahl.pdf.
14. encyclopedia2.thefreedictionary.com
15. linustechtips.com
16. youtube.com/watch?v=w3K1JklY6D4