10-MA’RUZA. OPERATSION TIZIMLARDA XOTIRANI BOSHQARISH

(Xotira va uning aksi, virtual address makoni. Xotirani boshqarishning umumiy tamoyillari, xotirani static va dinamik bo'limlar bo'yicha taqsimlash, segmentli, sahifali, segment-sahifali tashkil etish)

Kompyuter xotirasi turlari va ularning tashkil etilishi. Xotira aniq boshqaruvni talab qiladigan juda muhim resurs. Asosiy xotirani boshqarish juda muhimdir. Aslida, tarixiy nuqtai nazardan, butun tizimning ishlashi to‘g‘ridan-to‘g‘ri ikkita narsaga bog‘liq edi: ishlov berishda qancha xotira mavjudligi va u qanday optimallashtirilganligida.

Xotira ierarxiyasini boshqaradigan operatsion tizimning qismi xotira menejeri yoki menejer deb ataladi. U xotirani samarali boshqarish uchun mo‘ljallangan va xotiraning qaysi qismlari ishlatilishini kuzatishi, jarayonlar uchun kerak bo‘lgan xotirani ajratishi va jarayonlar o‘z ishini tugatgandan so‘ng xotirani bo‘shatishi kerak.

Xotira menejeri (shuningdek, operativ xotira yoki RAM, yoki asosiy xotira sifatida ham tanilgan) va xotirani taqsimlash sxemalarining to‘rtta turi mavjud: bir foydalanuvchili tizimlar, belgilangan qismlar, dinamik qismlar va ko‘chiriladigan dinamik qismlar.

Hozirgi kundagi zamonaviy operatsion tizimlarda virtual xotira tushunchasi mavjud. Xotira – bu ko‘p dasturli operatsion tizim tomonidan ehtiyotkorlik bilan boshqarishni talab qiladigan muhim resursdir. Xotira osti sifatida bu yerda tezkor (operativ) xotira tushuniladi. Tashqi xotira (saqlash qurilmasi) deb nomlanadigan qattiq diskning xotirasidan farqli o‘laroq, ma’lumotlarni saqlash uchun operativ xotira doimiy quvvat manbai talab qiladi. Dastlabki operatsion tizimlarda xotirani boshqarish dasturni va uning ma’lumotlarini tashqi diskdan (perfokarta, magnit lenta yoki magnit disk) xotiraga yuklanar edi.

Ko‘p dasturlashni paydo bo‘lishi bilan, operatsion tizim bir vaqtning o‘zida bir nechta ishga tushuriladigan dasturlar o‘rtasida mavjud xotirani taqsimlash bilan bog‘liq yangi vazifalarga duch keldi.

Multidasturli tizimda xotirani boshqarish uchun OT funksiyalari quyidagilardan iborat:

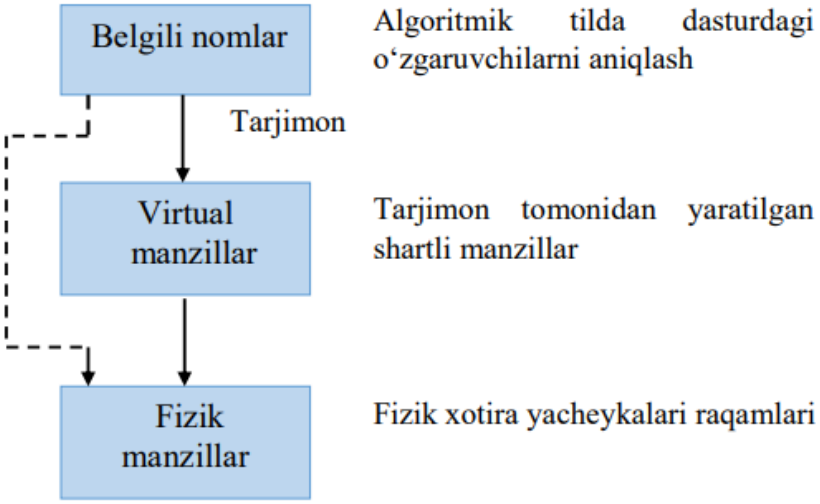
- bo‘sh va band xotirani kuzatish;
- jarayonlarga xotirani taqsimlash va jarayonlar tugaganda xotirani bo‘shatish;
- asosiy xotira o‘lchamlari undagi barcha jarayonlarni joylashtirish uchun yetarli bo‘lmaganda jarayon kodlari va ma’lumotlarini operativ xotiradan diskka ko‘chirish (to‘liq yoki qisman), va qachonki unda bo‘sh joy paydo bo‘lganda ularni RAM ga qaytarish;
- fizik xotiraning ma’lum bir sohasiga dastur manzillarini o‘rnatish;
- xotirani himoya qilish - boshqa jarayonga taqsimlangan xotiraga ma’lumotlarni yozishni yoki u yerdan o‘qishni taqiqlash. Ushbu funksiya qoida tariqasida operatsion tizim dasturiy modullari tomonidan qurilma vositalari bilan yaqin hamkorlikda amalga oshiriladi.

Kompyuter xotirasining fizik tuzilishi (tashkil etilishi) Kompyuterning xotira qurilmasi ikki xil turga: asosiy (bosh xotira, tezkor xotira, fizik xotira) va ikkilamchi (doimiy xotira, ichki xotira) xotiraga bo‘linadi. Asosiy xotira bir baytli tartiblangan yacheyka massiviga ega bo‘lib, har bir yacheyka o‘zining manziliga (raqamiga) ega. Protsessor buyruqlarni asosiy xotiradan oladi, qayta ishlaydi va bajaradi. Buyruqlarni bajarishda asosiy xotiraning bir nechta yacheykalariga murojaat qilishga to‘g‘ri keladi.

Odatda asosiy xotira yarim o‘tkazgichli texnologiya asosida tayyorlanadi, shuning uchun xotiradagi ma’lumotlar elektr manbasidan uzilgandan so‘ng o‘chib ketadi. Ikkilamchi xotira (bu asosan qattiq disklardir) bu chiziqli birlik manzilga ega bo‘lgan joy va ularni ketma-ket joylashgan baytlar tashkil qiladi. Ikkilamchi xotiraning tezkor xotiradan farqi shundaki, u alohida energiyaga, katta hajmga, va samarali foydalanish imkoniyatiga ega. 3.1-rasmdagi ko‘rsatilgan sxemaga yana bir nechta oraliq satxlarni qo‘shish mumkin. Har xil ko‘rinishdagi xotiralar ierarxiyaga, murojaat vaqti kamayib borishi, narxini oshishi va sig‘imi oshishi tarzida birlashishi mumkin.

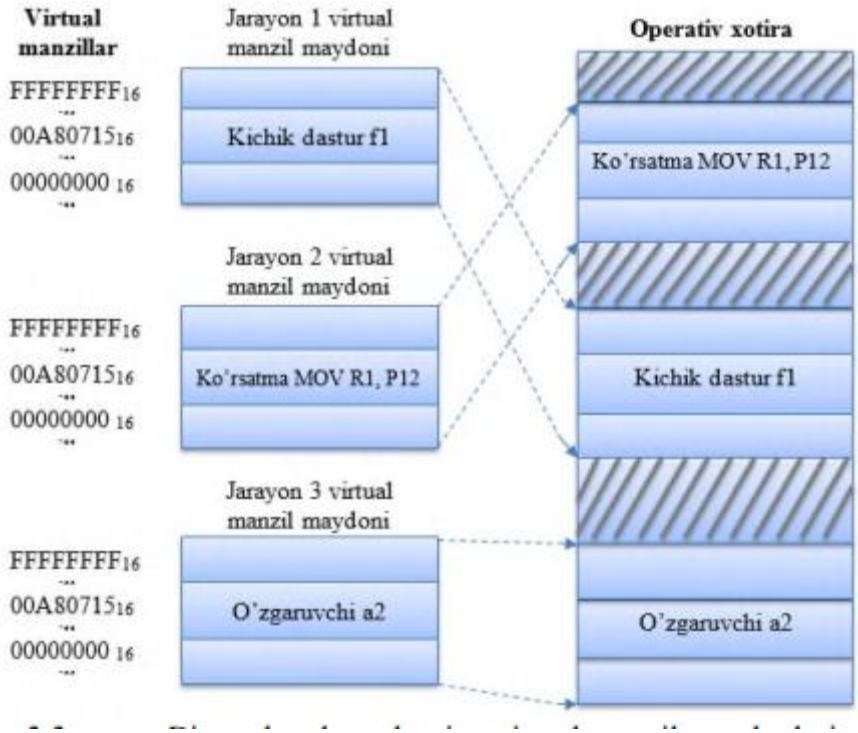
3.1- rasm. Xotira ierarxiyasi

Ko‘p bosqichli sxemalar quyidagicha ishlatiladi. Ma’lumotlar odatda xotiraning yuqori satxlaridan qidiriladi, agar u yerdan topilmasa, ma’lumotlar katta raqamli satxlarda ham saqlanadi. Shuning uchun, u keyingi satxdan qidira boshlaydi. Agar kerakli ma’lumotni topsa, uni yuqoriroq satxga o‘tkazadi. Manzil turlari O‘zgaruvchilar va buyruqlarni aniqlash uchun dastur hayot siklining turli bosqichlarida belgili nomlar (belgilar), virtual manzillar va fizik manzillar ishlatiladi (3.2- rasm). Belgili nomlar – foydalanuvchi tomonidan dasturni algoritmik tilda yoki assemblerda yozishda belgilanadi. Virtual manzillar (matematik yoki mantiqiy) – dasturni mashina tiliga tarjima qiladigan tarjimon tomonidan yaratiladi. Tarjima paytida dastur tezkor xotiraning qayeriga yuklanishi umuman ma’lum emas, chunki tarjimon odatda o‘zgaruvchilar va buyruqlarga shartli virtual manzillarni tayinlaydi, odatda dasturning boshlang‘ich manzili nol manzili bo‘ladi. Fizik manzillar – o‘zgaruvchilar va buyruqlar joylashgan yoki joylashishi kerak bo‘lgan xotira xujayralari raqamlariga mos keladi. Barcha jarayonlar uchun mumkin bo‘lgan virtual maydon manzillari diapazoni bir xil. Masalan, 32-razryadli virtual manzillardan foydalanganda, bu oraliq 0000000016 va FFFFFFFF16 chegaralari bilan belgilanadi. Dastur tomonidan yaratilgan barcha virtual manzillarning to‘plami virtual manzil maydoni deb nomlanadi. Ushbu virtual manzillarga mos keladigan barcha fizik manzillarning to‘plami fizik manzillar maydoni deb nomlanadi. Virtualdan fizik manzilgacha ish vaqtini xaritalash, qurilma vositasi bo‘lgan xotirani boshqarish bloki (MMU) tomonidan amalga oshiriladi. MMU virtual manzilni fizik manzilga tarjima qilish uchun quyidagi mexanizmdan foydalanadi. Baza registridagi qiymat foydalanuvchi jarayoni natijasida hosil bo‘lgan har bir manzilga qo‘shiladi, ular xotiraga yuborilganida hisobga olinadi. Masalan, agar bazaviy registr qiymati 10000 bo‘lsa, foydalanuvchi 100 manzil manzilidan foydalanishga urinsa, dinamik ravishda 10100 manziliga joylashtiriladi. Foydalanuvchi dasturi virtual manzillar bilan shug‘ullanadi, u hech qachon haqiqiy fizik manzillarni ko‘rmaydi.



3.2- rasm. Manzil turlari

Shu bilan birga, har bir jarayon o‘zining virtual manzil maydoniga ega - tarjimon har bir dasturning parametrlari va kodlariga virtual manzillarni mustaqil ravishda tayinlaydi (3.3- rasm). Belgili nomlar Virtual manzillar Fizik manzillar Tarjimon Algoritmik tilda dasturdagi o‘zgaruvchilarni aniqlash Tarjimon tomonidan yaratilgan shartli manzillar Fizik xotira yacheykalari raqamlari



3.3- rasm. Bir nechta dasturlarning virtual manzil maydonlari

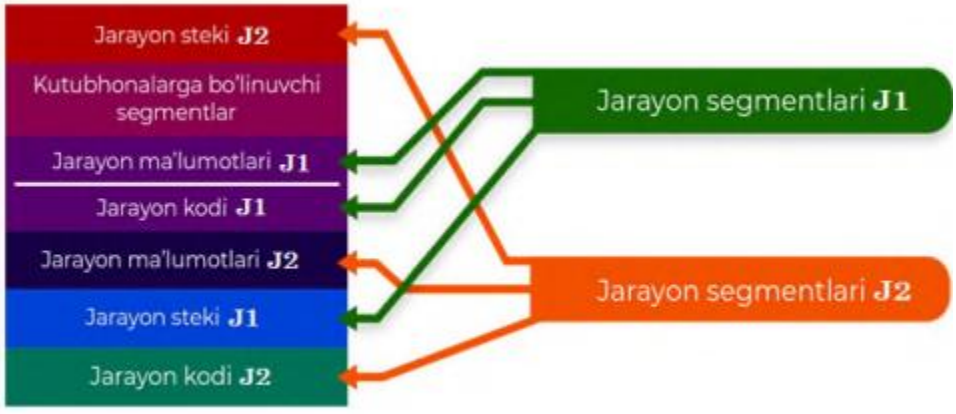
O‘zgaruvchining virtual manzillari va turli xil jarayonlar ko‘rsatmalarining bir-biriga to‘g‘ri kelishi nizolarga olib kelmaydi, chunki ushbu o‘zgaruvchilar bir vaqtning o‘zida xotirada bo‘lganida, operatsion tizim ularni turli xil fizik manzillarga joylashtiradi. Turli xil operatsion tizimlar virtual manzillar maydonini tuzishning turli usullaridan foydalanadilar. Ular chiziqli va segmentli. Ba’zi bir operatsion tizimlarda, fizik xotira singari, jarayonning virtual manzillar maydoni doimiy ravishda virtual manzillarning ketma-ketligi sifatida taqdim etiladi. Manzil maydonining bunday tuzilishi chiziqli deb nomlanadi.

Boshqa operatsion tizimlarda virtual manzillar maydoni segmentlar (yoki bo‘limlar, yoki maydonlar yoki boshqa atamalar) deb nomlangan qismlarga bo‘linadi. Fizik va mantiqiy xotira Fizik xotira. Ma’lum bo‘lishicha, bu usulda boshqarishni tashkil etish xotira satxlariga kirishni va aloqa chastotasini kamaytiradi. Bu yerda muhim rolni, chegaralangan vaqt davomida, xotira manzillarining kichik bo‘lagi bilan ishlash xossasi o‘ynaydi.

Protsessor keshi, qurilmalarning bir qismi hisoblanadi, shuning uchun operatsion tizimning xotira menejeri, asosan ma’lumotlarni kompyuterning asosiy va ichki xotira qismiga taqsimlash bilan shug‘ullanadi. Bazi sxemalarda tezkor va ichki xotira o‘rtasidagi oqimni dasturchi boshqaradi. Ammo bu bog‘lanish dasturchi vaqtini yo‘qotadi, shu sababli bu ishni OT ga yuklashga harakat qilinadi. Fizik xotirada ma’lumotlarni real joylashishini ko‘rsatuvchi asosiy xotiradagi manzillar - fizik manzillar deb ataladi. Dastur ishlaydigan fizik manzillar to‘plami, fizik manzillar maydoni deb ataladi.

Mantiqiy (logical) xotira. Xotirani, yacheykalar chiziqli to‘plami ko‘rinishida tashkil etish, dasturchining dastur va ma’lumotlar saqlanishi ko‘rinishi haqidagi tasavvuri bilan mos kelmaydi. Bazan jarayon tarkibiga kiruvchi hamma modullar xotirada ketma-ket joylashadi va chiziqli manzillar maydonini tashkil qiladi. Biroq ko‘pincha modullar xotiraning turli joylarida joylashtiriladi va turlicha foydalaniladi. Xotirani bosharish sxemasida, foydalanuvchining bunday tassavuriga mos keladigan ma’lumot va dasturlarni saqlash, segmentatsiya deyiladi. Segment-xotiraning aniq ko‘rsatilgan qismi bo‘lib, uning ichki qismida chiziqli manzillarni qo‘llab-quvvatlaydi. Segment protsedura, massiv, stek yoki skalyar miqdorlardan tashkil topgan bo‘ladi, lekin, odatda aralash turdagi ma’lumotlardan iborat bo‘lmaydi. Boshida segmentlar dastur kodi fragmentlarini (matn redaktori, trigonometrik kutubxona va x.k.) jarayonlar bilan umumlashtirish zaruriyatidan kelib chiqqan bo‘lishi kerak, chunki ularsiz har bir jarayon o‘zining manzil maydonida ma’lumotlarning yana bir nusxasini saqlashiga to‘g‘ri kelar edi.

Xotiraning, tizim bir nechta jarayonning ma’lumotlarini aks ettiradigan alohida qismlari bo‘lib ular segmentlar deb nom oldi. Xotira shunday qilib, chiziqli ko‘rinishdan ikki o‘lchamli ko‘rinishga keldi. Manzil ikki komponentdan iborat bo‘lib, ular: segment raqami va segment ichidagi joylashgan o‘rnidir. Keyinchalik, jarayonning turli komponentalarini (dastur kodi, ma’lumotlar, stek va x.k.) turli segmentlarda joylashtirish qulay bo‘lib qoldi. Yana shu narsa aniq bo‘lib qoldiki, aniq segment ishini, unga segmentda saqlanadigan ma’lumotlar ustida bajarilishi ruhsat berilgan operatsiyalar, masalan, murojaat xuquqi va operatsiyalar turi kabi atributlar qiymatini berib, nazorat qilish mumkin bo‘lib qoldi. Jarayon segmentlarining kompyuter xotirasida joylashishi 3.4- rasmda ko‘rsatilgan.



3.4- rasm. Jarayon segmentlarining kompyuter xotirasida joylashishi

Bazi, jarayonni manzil manzilini tasvirlaydigan segmentlar 3.4- rasmda ko‘rsatilgan. Aksariyat zamonaviy operatsion tizimlar xotirani segmentli boshqaruv xususiyatiga ega. OT larning bazi arxitekturalarida (masalan, Intel) segmentlash qurilmalar tomonidan qo‘llanadi. Jarayon murojaat qiladigan manzillar, operativ xotirada mavjud bo‘lgan real manzillardan shu

tarzda farq qiladi. Har bir aniq holatda dastur foydalanadigan manzil, har xil usullar yordamida tasvirlanishi mumkin. Masalan, manzil, berilgan matnda odatda belgili bo‘ladi. Kompilyator bu belgili manzil va o‘zgaradigan manzillarni bog‘laydi (masalan, n bayt modul boshidan). Dastur generatsiyalagan bunday manzil odatda mantiqiy manzil (virtual xotirali tizimlarda u ko‘pincha virtual xotira) deb nomlanadi. Barcha mantiqiy manzillar to‘plami mantiqiy (virtual) manzillar maydoni deb ataladi. Manzillar bog‘lanishi Demak, mantiqiy va fizik manzillar maydonlari, tashkil etilishi va o‘lchami bo‘yicha bir biriga mos emas. Mantiqiy manzillar maydoning maksimal o‘lchami odatda protsessorning razryadi bilan aniqlanadi (masalan 232), va zamonaviy tizimlarda fizik manzillar maydonining hajmidan ko‘zga ko‘rinarli darajada yuqori bo‘ladi. Shunday ekan, protsessor va operatsion tizim asosiy xotirada joylashgan dasturni dastur kodiga, real fizik manzilga tayangan holda yo‘lni aks ettirishi kerak. Bunday ko‘rinishda manzillarni tasvirlash manzillarni translatsiyasi yoki manzillarni bog‘lash deb nomlanadi. Mantiqiy manzilning fizik manzil bilan bog‘lanishi dastur operatorining bajarilishigacha yoki bajarilish vaqtida amalga oshirilishi shart. Bunday holda, ko‘rsatmalarni va ma’lumotlarni xotiraga bog‘lash quyidagi qadamlar bo‘yicha amalga oshiriladi.

- Kompilyatsiya bosqichi. Kompilatsiya bosqichida siz jarayonning xotirada joylashish joyini bilsangiz, unda mutloq (absolyut) kodni yaratishingiz mumkin. Masalan, agar foydalanuvchi jarayoni doimiy ravishda R joyida boshlanishini bilsangiz, u holda yaratilgan kompilyator kodi o‘sha joyda boshlanadi va u yerdan davom etadi. Agar bir muncha vaqt o‘tgach, dastlabki joylashuv o‘zgarsa, ushbu kodni qayta yozishingiz kerak bo‘ladi.
- Yuklash bosqichi. Agar kompilatsiya bosqichida dasturlarni joylashtirish to‘g‘risida axborot mavjud bo‘lmasa, u holda kompilyator o‘zgaruvchan kodni yaratishi kerak. Bu holatda yakuniy bog‘lanish yuklanish vaqtigacha xotirada saqlanadi. Agar dastlabki manzil o‘zgarsa, o‘zgarishlarni hisobga olib foydalanuvchi kodini qayta yuklash kerak bo‘ladi.
- Bajarilish bosqichi. Agar jarayonni bajarilish vaqtida bir xotira segmentidan boshqa xotira segmentiga o‘tkazish mumkin bo‘lsa, u holda bajarilish muddati tugaguncha bog‘lanish kechiktirilishi kerak. Bu yerda mavjud maxsus qurilmalardan foydalangan ma’qul, masalan, register joyini o‘zgartirish. Ko‘plab zamonaviy operatsion tizimlar bajarish bosqichida manzil translatsiyasidan foydalanadi, buning uchun maxsus qurilma vositalaridan foydalaniladi.

Xotirani boshqarish va undan umumiy foydalanish mexanizmi Xotiraning boshqaruv tizimi funksiyasi Xotiradan samarali foydalanishni ta’minlash uchun operatsion tizim quyidagi funksiyalarni bajarishi lozim:

- Fizik xotirani aniq bir sohasida jarayon manzillar to‘plamini aks ettirish;
- Qarama-qarshi jarayonlar o‘rtasida xotirani taqsimlash;
- Jarayonlar manzillar maydoniga ruhsatni boshqarish;
- Operativ xotirada joy qolmaganda, tashqi xotiraga jarayonlarni (qisman yoki to‘liq) yuklash;
- Bo‘sh va band xotirani hisobga olish. Bir foydalanuvchili sxema Dastlabki operatsion tizimlarda xotirani boshqarishning eng oddiy usullari qo‘llanilgan. Boshida foydalanuvchining har bir jarayoni asosiy xotiraga ko‘chirilishi kerak bo‘lgan, xotira uzluksiz maydonini band qilgan, tizim esa qo‘shimcha foydalanuvchi jarayonlarga bir vaqtning o‘zida asosiy xotirada joylashib turgunicha xizmat ko‘rsatadi. Xotirani taqsimlashning birinchi sxemasi: 3.5-rasmda ko‘rsatilgandek, har bir qayta ishlangan dastur to‘liq xotiraga yuklandi va unga zarur bo‘lgan xotira maydonini ajratdi. Bu yerda kalit so‘zlar yaxlitlik va doimiylikdir. Agar dastur juda katta bo‘lsa va mavjud xotira maydoniga mos kelmasa (xotira maydoni o‘lchami yetarli bo‘lmasa), uni bajarib bo‘lmaydi. Va dastlabki kompyuterlar juda katta bo‘lishiga qaramay, ular juda kam xotiraga ega edilar. 96 3.5- rasm. Bitta dastur bir vaqtning o‘zida xotirada saqlanadi. Xotiraning qolgan qismi ishlatilmaydi Bu barcha kompyuterlar uchun sezilarli cheklovni

namoyish etadi – ular faqat cheklangan xotiraga ega va agar dastur xotira maydoniga mos kelmasa asosiy xotiraning hajmini oshirish kerak yoki dasturni o‘zgartirish (kichraytirish) kerak. Odatda, uni kichiklashtirish yoki dastur segmentlarini (dasturga bo‘laklarni) to‘ldirishga imkon beradigan usullardan foydalanish orqali o‘zgartiriladi. (Qoplash usuli (overlay) - bu dastur segmentlarini ikkinchi darajali xotiradan asosiy xotiraga bajarish uchun o‘tkazishdir, shunda ikki yoki undan ortiq segmentlar bir xil xotira maydonlarini egallab turgan navbatni egallab olishlari mumkin).

Overlay - bu tizimda o‘rnatilganidan ko‘ra ko‘proq xotirani egallaydigan dasturlarni yaratishga imkon beradigan dasturlash usulidir. Har bir foydalanuvchiga har bir vazifa uchun mavjud bo‘lgan asosiy xotiradan foydalanish huquqi beriladi va vazifalar ketma-ket bajariladi. Xotirani taqsimlash uchun operatsion tizim oddiy algoritmdan foydalanadi (muammoni hal qilish uchun qadamma-qadam protsedura): Vazifani bir foydalanuvchi tizimiga yuklash algoritmi

1. Dastur xotirasini asosiy registrda saqlash (xotirani himoyasi uchun)
2. Dastur hisoblagichini birinchi xotira yacheyasining manziliga o‘rnatish (u dastur tomonidan ishlatiladigan xotira miqdorini kuzatadi)
3. Dasturning birinchi ko‘rsatmasini o‘qish
4. Dastur hisoblagichini ko‘rsatmadagi baytlar soniga ko‘paytirish
5. So‘nggi ko‘rsatma berildimi? agar ha bo‘lsa, dasturni yuklashni to‘xtatish agar yo‘q bo‘lsa, unda 6-bosqichdan davom etish
6. Dastur hisoblagichi xotira hajmidan kattaroqmi? agar ha bo‘lsa, dasturni yuklashni to‘xtatish agar yo‘q bo‘lsa, unda 7-bosqich bilan davom etish
7. Ko‘rsatmani xotiraga yuklash
8. Keyingi dastur ko‘rsatmalarini o‘qish
9. 4-bosqichga o‘tish.

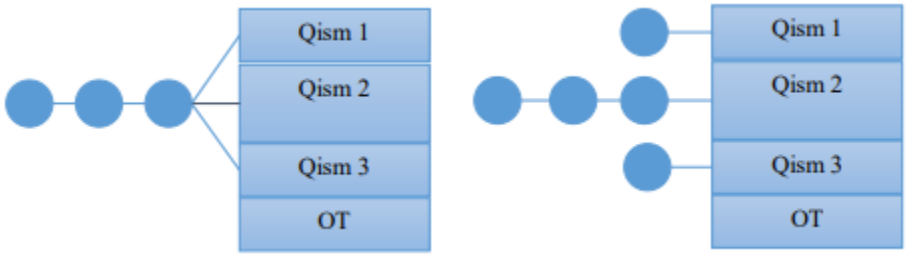
E’tibor bering, operatsion tizimning “Xotira menejeri” tomonidan bajarilgan ishlarning hajmi minimal, funksiyalarni bajarish kodi sodda va mantiq juda oddiy. Faqat ikkita qurilma vositasi kerak bo‘ladi: asosiy manzilni saqlash uchun registr va xotirada o‘qilayotgan paytda dastur hajmini kuzatib borish uchun akkumulyator (y ham register). Dastur xotiraga to‘liq yuklanganidan so‘ng, u odatiy tarzda yoki operatsion tizimning aralashuvi bilan tugatilmaguncha o‘sha yerda qoladi. Ushbu turdagi xotirani taqsimlash sxemasining asosiy muammolaridan biri shundaki, u ko‘p dasturlash yoki tarmoqni qo‘llab-quvvatlamaydi, u bir vaqtning o‘zida faqat bitta vazifani bajara oladi. Bu usul 1940- va 1950- yillardagi tijorat kompyuterlarida qo‘llanilgan. Shuning uchun, 1950- yillarning ohiri va 1960- yillarning boshlarida xotirani boshqarishning yangi sxemasi talab qilindi, va mustaqil operatsiyalarni takrorlash orqali kompyuter tizimi resurslaridan foydalanishga imkon beradigan qismlardan foydalanildi. Xotirani boshqarishning ushbu sxemalari bugungi operatsion tizimlar tomonidan kam ishlatiladi, ammo o‘rganish juda muhimdir, chunki har biri xotira boshqaruvini rivojlantirishga yordam beradigan asosiy tushunchalarni taqdim etadi. Kompyuter tizimi bosh asosiy masalasi dasturni boshqarishdir. Xotiraning alohida roli protsessor dastur ko‘rsatmalarini ular xotirada bo‘lgan taqdirdagina bajarishi mumkinligi bilan izohlanadi. Dasturlar va ularning murojaat qiladigan ma’lumotlari, bajarilish jarayonida operativ xotirada (hech bo‘lmasa qisman) joylashgan bo‘lishi shart. Operatsion tizimga, xotirani, foydalanuvchi jarayonlari va OT komponentalari orasida taqsimlashga to‘g‘ri keladi.

Operatsion tizimning bu faoliyati xotirani boshqarish deyiladi. Shunday qilib, xotira sinchiklab boshqarishni talab etadigan resursdir. Yaqin kunlargacha xotira eng qimmat resurs hisoblangan. Operatsion tizimning, xotirani boshqaradigan qismi, xotira menejeri deyiladi. Xotira amaliy dastur modullari va operatsion tizimning modullari o‘rtasida ham taqsimlanadi.

Xotirani taqsimlash, qayta taqsimlash va jamlash usullari. Belgilangan (statik) qismlar Tezkor xotirani boshqarishning eng oddiy yo‘li uni oldindan (ishga tushurish bosqichida yoki tizim yuklanishi vaqtida) bir qancha belgilangan (statik) o‘lchamdagi qismlarga bo‘lishdan iboratdir. Kelib tushayotgan jarayonlar u yoki bu qismga joylashtiriladi. Shu sababli fizik manzillar

maydonining shartli bo‘linishi yuzaga keladi. Jarayonning mantiqiy va fizik manzillari bog‘lanishi uni aniq bir qismga yuklash vaqtida yoki bazan kompilatsiya vaqtida yuzaga keladi. Har bir qism o‘zining jarayonlar navbatiga ega, yoki hamma qismlar uchun jarayonlar global navbati mavjud bo‘lishi mumkin. Bu sxema IBM OS/360 (MFT), DES RSX-11 va shunga yaqin boshqa tizimlarda qo‘llanilgan.

Xotirani boshqarish tizimi jarayonni hajmini baholaydi, unga mos keluvchi qismni tanlaydi, jarayonni bu qismga yuklaydi va manzillarni sozlaydi. 3.6- rasmda belgilangan qismli sxemalar ko‘rsatilgan: (a) navbati umumiy bo‘lgan jarayonlar, (b) alohida navbatli jarayonlar. Bu sxemaning kamchiligi ko‘rinib turibdiki, bir vaqtda bajariladigan jarayonlar soni qismlar soni bilan cheklangan. Boshqa muhim kamchiligi shundan iboratki, taklif qilinayotgan sxema, ichki fragmentlashdan, yani jarayonga ajratilgan, ammo ishlatilmagan xotira qismini yo‘qotish bilan qattiq zararlanadi. Fragmentatsiya, jarayon o‘ziga ajratilgan qismni to‘liq band qilmasligi yoki bazi qismlar, bajariladigan foydalanuvchi dasturlari uchun kichik bo‘lganligidan kelib chiqadi.



3.6- rasm. Overleyli (qoplangan) tuzilish

Jarayon mantiqiy manzillar maydoni hajmi, unga ajratilgan qism hajmidan katta (yoki eng katta hajmdan ham katta) bo‘lgan holatlarda, ba’zan overley nomli (yoki qoplanadigan tuzilishli) tashkil etadigan texnikadan foydalaniladi. Qism 1 Qism 2 Qism 3 OT Qism 1 Qism 2 Qism 3 OT 99 Asosiy g‘oya – faqat ayni vaqtda kerak bo‘lgan dastur ko‘rsatmalarini xotirada saqlab turishdir. Overley tuzilishning tavsifini yozish uchun odatda maxsus sodda (overley description language) tildan foydalaniladi.

Asosiy xotirada qo‘llanilgan belgilangan qismlar (statik qismlar deb ham ataladi) multidasturlashga imkon beradigan birinchi urinishdir – har bir vazifa uchun bitta qism. Har bir qismning o‘lchami tizim ishga tushurilganda belgilanganligi sababli, har bir qism faqat kompyuter tizimi o‘chirilgan, qayta konfiguratsiya qilingan va qayta ishga tushirilganida qayta tuzilishi mumkin edi. Shunday qilib, tizimni ishga tushirgandan so‘ng, qism o‘lchamlari o‘zgarishsiz qoldi. Ushbu sxemada muhim omil paydo bo‘ldi: vazifa xotirasi maydonini himoya qilish. Qismga vazifa berilgandan so‘ng, boshqa vazifalarga unga tasodifan yoki qasddan kirishga ruhsat berilmagan. Qismlarga kirishning bu usuli bir foydalanuvchili tizimga qaraganda har bir qismni himoya qilishni nazarda tutadi. Chunki bir foydalanuvchilik tizimda istalgan vaqtda faqat bitta ish asosiy xotirada bo‘lgan, shuning uchun operatsion tizimning asosiy xotirada joylashgan qismi himoya qilinishi kerak edi. Shu bilan birga, belgilangan qismni taqsimlash sxemalari uchun asosiy xotirada mavjud bo‘lgan har bir qism uchun himoya majburiy edi.

Odatda bu kompyuter qurilmalari va operatsion tizimning birgalikdagi javobgarligi edi. Vazifalarni xotirada saqlash uchun ishlatiladigan algoritm bir foydalanuvchi tizimida ishlatilgandan ko‘ra bir necha marotaba ko‘proq qadamlarni talab qiladi, chunki vazifa hajmi to‘liq mos kelishiga ishonch hosil qilish uchun bo‘lim hajmiga mos kelishi kerak. Keyin, yetarlicha kattalikdagi blok aniqlanganda, uning mavjudligini tekshirish uchun qismning holatini tekshirish kerak.

Har bir qism faqat bitta dastur tomonidan ishlatilishi mumkin. Har bir qismning o‘lchamini kompyuter operatori oldindan belgilab qo‘ygan, shuning uchun tizimni qayta ishga tushurmasdan hajmni o‘zgartirish mumkin emas. Vazifani belgilangan qismlarga yuklash algoritmi

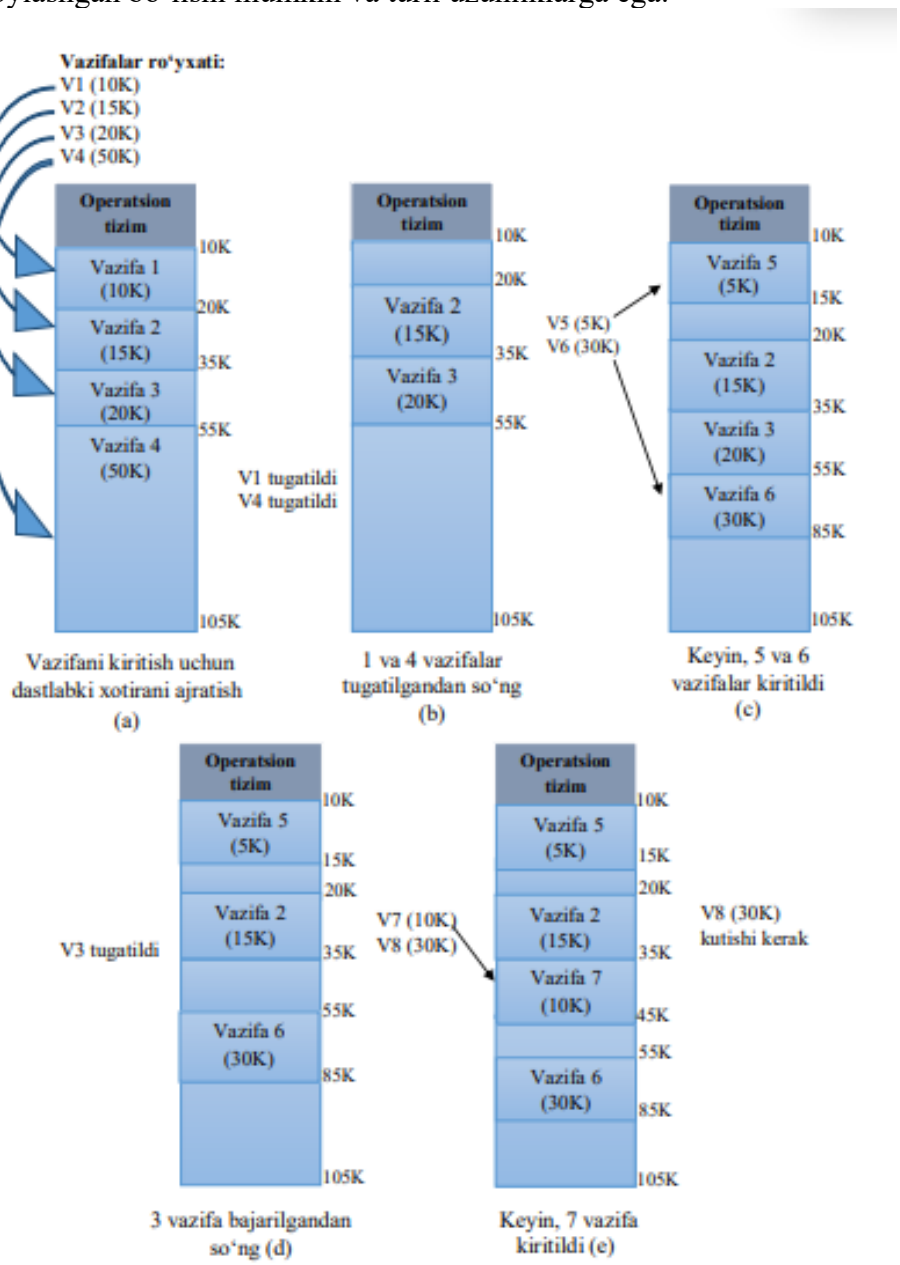
1. Vazifa talab qilgan xotira hajmini aniqlash

2. Agar vazifa hajmi > qism hajmidan katta bo‘lsa Unda, vazifani rad etadi Operatorga kerakli xabarni chop etadi Navbatdagi vazifani bajarish uchun 1-qadamga o‘tish Aks holda 100 3-qadamdan davom etadi
3. Hisoblagichni 1 ga o‘rnatish
4. Hisoblagich o‘rnatilganda xotiradagi bo‘limlar soni katta yoki teng bo‘lganda bajarish Agar vazifa hajmi xotira qismi hajmidan (hisoblagich) katta bo‘lsa Unda hisoblagich = hisoblagich + 1 Aks holda Agar xotira qismi hajmi (hisoblagich) = “bo‘sh” bo‘lsa Unda xotira qismiga (hisoblagich) vazifani yuklash Xotira qismi holatini (hisoblagich) “band” ga o‘zgartirish Aks holda Hisoblagich = hisoblagich + 1 Bajarishni tugatish
5. Hozirgi vaqtda bo‘sh qismlar mavjud bo‘lmasa, vazifani kutish navbatiga qo‘yish
6. Navbatda turgan vazifani bajarish uchun 1-qadamga o‘tish. Ushbu taqsimlash sxemasi bir foydalanuvchi tizimiga qaraganda ancha moslashuvchan, chunki u bir vaqtning o‘zida bir nechta dasturlarni xotirada saqlashga imkon beradi. Biroq, bu hali ham dasturni doimiy ravishda va bajarilishning boshidan oxirigacha xotirada saqlashni talab qiladi. Vazifa uchun xotira maydonini taqsimlash uchun operatsion tizimning xotira menejeri har bir xotira qismining o‘lchamini, uning manzili, kirish cheklovlari va tizimning joriy holatini (bo‘sh yoki band) ko‘rsatadigan jadvalni (3.1- jadval) saqlashi kerak. 3.1- jadval Qism hajmi Xotira manzili Ruxsat Qism holati 100 K 200 K Vazifa 1 Band 25 K 300 K Vazifa 4 Band 25 K 325 K Bo‘sh 50 K 350 K Vazifa 2 Band 101 Har bir vazifa bajarilib tugatilgach, uning xotira qismining holati band holatdan bo‘shga o‘zgaradi, va ushbu qismga kiruvchi boshqa vazifani kiritish mumkin. 3.7- rasm.

Asosiy xotiradan qismlarga ajratilgan holda foydalanish Asosiy xotiradan qismlarga ajratilgan holda foydalanish 3.1- jadvalda keltirilgan. Vazifa 1 mavjud 100 Kbayt xotiraning atigi 30 Kbaytini oladi, vazifa 3 qismlarni bo‘shashini kutishi kerak, hatto 1- qismda 70K bo‘sh joy mavjud bo‘lsa ham. Vazifalarga “kerakli hajmdagi birinchi mavjud qism” asosida bo‘sh joy ajratilgan. Agar tizimda bajariladigan barcha vazifalar bir xil o‘lchamli bo‘lsa yoki o‘lchamlari oldindan ma’lum bo‘lsa va konfiguratsiyaga qarab o‘zgarmasa, belgilangan qismlar sxemasi yaxshi ishlaydi. Ideal holda, bu tizimda yaqin soatlarda, kunlarda yoki haftalarda bajariladigan barcha vazifalar to‘g‘risida aniq ma’lumot talab qiladi. Ammo, agar operator kelajakni aniq taxmin qila olmasa, qismlarning o‘lchamlari tasodifiy ravishda belgilanadi va ularning o‘lchami kiruvchi vazifalar uchun juda kichik yoki juda katta bo‘lishi mumkin. Qismning o‘lchamlari juda kichik bo‘lsa, jiddiy oqibatlarga olib keladi. Boshqa tomondan, agar qismlar juda katta bo‘lsa, xotira behuda sarflanadi. Agar vazifa to‘liq qismni egallamasa, qismda foydalanilmagan xotira bo‘sh qoladi, uni boshqa vazifaga o‘tkazish mumkin emas, chunki har bir qism bir vaqtning o‘zida bitta vazifaga beriladi. Bu ajralmas birlikdir. 3.7- rasmda shunday vaziyatlardan biri ko‘rsatilgan. Ruxsat etilgan qismlardan qisman foydalanish va bir vaqtning o‘zida qism ichida foydalanilmagan bo‘shliqlarni yaratishning bu hodisasi ichki bo‘linish (fragmentatsiya) deb ataladi 4-qismda vazifa 2 (50K) 2-qismda vazifa 4 (25K) Asosiy xotira 1-qismda vazifa 1 (30K) Bo‘sh qism Ichki bo‘linish Qism 1 = 100K Qism 4 = 50K Qism 2 = 25K Qism 3 = 25K Asosiy xotira Vazifalar ro‘yxati Vazifa 1 = 30K Vazifa 2 = 50K Vazifa 3 = 30K (kutmoqda) Vazifa 4 = 25K 200K mavjud 102 va belgilangan qismlar uchun xotirani taqsimlash sxemasining asosiy kamchiligi hisoblanadi.

Dinamik qismlar. Dinamik qismlarda mavjud xotira qo‘shni bloklarda saqlanadi, lekin vazifalarga yuklanish vaqtida qancha xotira maydoni kerak bo‘lsa, shuncha xotira maydoni taqdim qilinadi. Bu belgilangan qismlarga nisbatan sezilarli yaxshilanish bo‘lsada, bu muammoni to‘liq hal qilmaydi. 3.8- rasmda ko‘rsatilgandek, dinamik qismlarni taqsimlash sxemasi birinchi vazifalarni yuklashda xotiradan to‘liq foydalanadi. Ammo tizimga yangi vazifalar kiritilganda, vazifaning kattaligi bo‘shagan xotirani hajmiga to‘g‘ri kelmasa, ular bo‘sh joyga ustuvorlik (prioritet) tartibida joylashtiriladi. 3.8- rasmda “birinchi kelganga - birinchi xizmat” ustuvorligi ko‘rsatilgan. Shunday qilib, keyingi xotirani taqsimlash, taqsimlangan xotira bloklari o‘rtasida bo‘sh xotiraning bo‘laklarini (fragmentlarini) yaratadi. Ushbu muammo tashqi bo‘linish deb

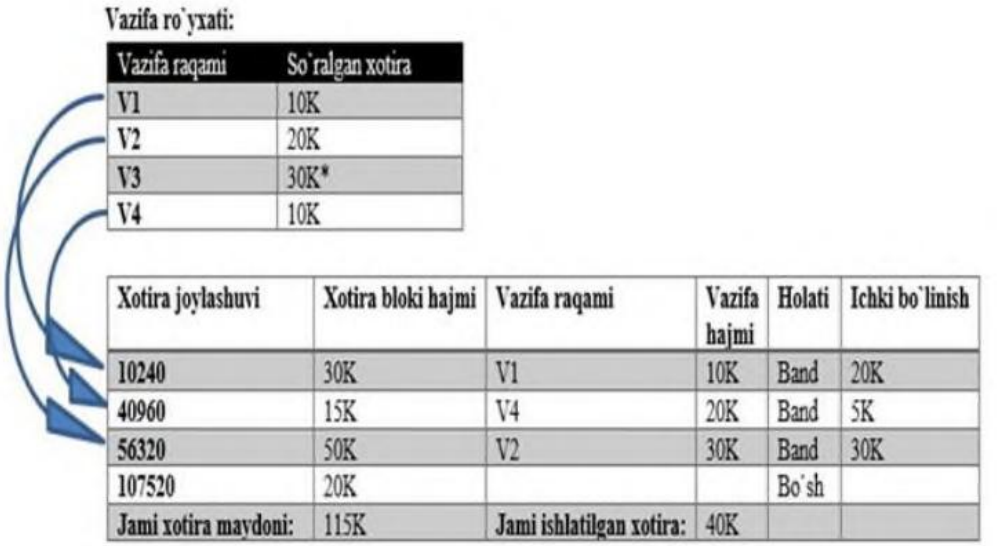
nomlanadi va ichki bo‘linish singari, xotirani yo‘qotishga imkon beradi. 3.8- rasmda (e) uchta bo‘sh qism mavjud 5 KB, 10 KB va 20 KB jami 35 KB, 8- vazifani bajarish uchun atigi 30 KB kerak bo‘ladi. Biroq, ular bir-biriga qo‘shni emasligi va vazifalar doimiy ravishda yuklanganligi sababli, ushbu sxema 8- vazifani kutishga majbur qiladi. Asosiy xotirani taqsimlashda dinamik qismlardan foydalanish. Quyidagi 5 ta rasmda (a-e) asosiy xotiraga qayta ishlash uchun 8 ta vazifa yuborildi va “birinchi kelganga - birinchi xizmat” algoritmi asosida xotira maydoni ajratilgan. Hatto qismlar orasida yetarli bo‘sh xotira bo‘lsa ham, 8 vazifa (e) kutishi kerak. Xotirani taqsimlashning umumiy vazifasi va uni hal qilish strategiyalari Umuman olganda, operatsion tizimlarda bir nechta qo‘shni hududlarda qo‘shni xotirani taqsimlash qo‘llanilishi mumkin. Bo‘sh maydon – bu bo‘sh xotiraning qo‘shni blokidir. Bo‘sh maydonlar tasodifiy ravishda xotiradan tarqalishi mumkin. Jarayonni yuklashda, uni joylashtirish uchun yetarlicha katta bo‘lgan har qanday bo‘sh qo‘shni hududdan xotira taqdim etiladi. Shu bilan birga operatsion tizim bo‘sh xotira maydonlari ro‘yhati va band xotira maydonlari ro‘yhatini saqlaydi. Ushbu maydonlarning barchasi tasodifiy xotirada joylashgan bo‘lishi mumkin va turli uzunliklarga ega.



3.8- rasm. Qismlarni dinamik taqsimlashda asosiy xotiradan foydalanish

Operatsion tizim mos qismni qidirmaydi, lekin yetarli hajmga ega bo‘lgan eng yaqin joylashgan xotira qismiga ishni taqsimlaydi. Ushbu qismlar birinchi mos keladigan xotirani

ajratish (birinchi qism talablariga javob beradi) yoki eng yaxshi (yo‘qotishlarning eng kam miqdori, talablarga javob beradigan eng kichik qism) xotira taqsimoti (Best-Fit Allocation) asosida ajratilishi mumkin. Ikkala sxemada ham xotira menejeri bo‘sh yoki foydalanilgan qismlarning (bo‘sh/band) xotira ro‘yxatlarini hajmiga yoki joylashishiga qarab tashkil qiladi. Eng yaxshi mos keladigan taqsimlash usuli bo‘sh/band bo‘lgan ro‘yxatlarni kichikdan kattasiga qarab tartiblaydi. Birinchi mos usul (first-fit method) xotira maydonlari tomonidan tashkil etilgan bo‘sh/band bo‘lmagan ro‘yxatlarni, past darajali xotiradan yuqori darajali xotiraga qadar saqlaydi. Ularning har biri ma’lum bir taqsimlash sxemasining ehtiyojlariga qarab o‘z afzalliklariga ega - eng yaxshi taqsimlash usuli odatda xotira maydonidan unumli foydalanishni ta’minlaydi, birinchi mos taqsimlash algoritmi tezroq taqsimlashni amalga oshiradi. Birinchi mos keladigan sxemadan foydalanib, 1- vazifa birinchi bo‘sh joyni talab qiladi. 2- vazifa, birinchi qismni talab qiladi, joylashishi uchun yetarlicha katta, ammo 3- vazifani bajarish uchun yetarlicha katta bo‘lgan oxirgi blokni talab qiladi. Shuning uchun, 3- vazifa (yulduzcha bilan belgilangan) 75 Mb foydalanilmagan xotira maydoni (ichki qism) mavjud bo‘lsa ham, katta blok paydo bo‘lguncha kutishi kerak. E’tibor bering, xotira ro‘yxati xotira maydoniga qarab tartiblangan.



3.9- rasm. Birinchi mos keladigan sxemadan foydalanish

Xotiradan foydalanish hajmi oshirildi, ammo xotirani taqsimlash jarayoni ko‘proq vaqtni talab etadi. Bundan tashqari, ichki bo‘linish kamaygan bo‘lsa ham, u to‘liq yo‘q qilinmadi. Birinchi mos keladigan algoritm, xotira menejeri uchun ikkita ro‘yxatni saqlaydi, birinchisi bo‘sh xotira bloklari va ikkinchisi band qilingan xotira bloklari. Operatsiya har bir vazifa hajmini har bir xotira blokining o‘lchami bilan mos keladigan yetarlicha katta blok topilmaguncha taqqoslaydigan oddiy sikldan iborat. Keyin vazifa ushbu xotira blokida saqlanadi va xotira menejeri keyingi navbatni kirish navbatidan olish uchun sikldan chiqadi. Agar butun ro‘yxat behuda qidirilsa, u holda vazifa kutish navbati ichiga joylashtiriladi. Keyin xotira menejeri keyingi vazifani tanlaydi va jarayonni takrorlaydi. Eng yaxshi taqsimlash sxemasi mos keladi. 1-vazifa 2 va 3- vazifalar kabi eng yaqin bo‘sh qismda taqsimlanadi. 4-vazifa eng mos bo‘lmasa ham, bo‘sh bo‘lgan yagona qismga taqsimlangan. Ushbu sxemada barcha to‘rtta vazifa kutishsiz qayta ishlanadi. Yodda tuting, xotira ro‘yxati xotira hajmiga qarab tartiblangan. Ushbu sxema yordamida xotiradan yanada samarali foydalaniladi, ammo uni amalga oshirish sekinroq hisoblanadi.

Eng mos va birinchi mos keladigan algoritmlar juda farq qiladi. Birinchi usul qanday amalga oshiriladi:

First-Fit Algoritmi

1. Hisoblagichni 1 ga o‘rnatish

2. Bajarishda, hisoblagich <= xotiradagi bloklar soni Agar vazifa hajmi > xotira hajmi (hisoblagich) bo‘lsa Unda, hisoblagich = hisoblagich + 1 Xotiraga (hisoblagich) vazifani yuklash Bo‘sh/band xotira ro‘yhatlarini sozlash 4-bosqichga o‘tish Tugatish
3. Vazifani kutish navbatiga qo‘yish
4. Keyingi vazifaga o‘tish. 3.2-jadvalda 200 bo‘sh maydonni bloklash so‘rovi faqat xotira menejeriga berilgan (maydonlar so‘zlar, baytlar yoki tizim boshqaradigan boshqa birlik bo‘lishi mumkin). Birinchi mos keladigan algoritmdan foydalanib va ro‘yxatning yuqorisidan boshlab, xotira menejeri 6785 manzilida joylashgan vazifani bajarishi uchun yetarlicha katta bo‘lgan birinchi xotira blokini topadi. Keyin, vazifa 6785 maydondan boshlanib, keyingi 200 bo‘sh maydonni egallaydi. Keyingi qadam, bo‘sh xotira bloki hozirda 6985 (6785 emas, balki 108 avvalgi kabi) maydonda joylashganligini va unda faqat 400 ta bo‘sh maydon mavjudligini (oldingidek 600 emas) belgilash uchun bo‘sh ro‘yxatni o‘rnatishdir.

So`rovdan oldin	
Boshlang`ich manzil	**Xotira bloki hajmi**
4075	105
5225	5
6785	600
7560	20
7600	205
10250	4050
15125	230
24500	1000

So`rovdan keyin	
Boshlang`ich manzil	**Xotira bloki hajmi**
4075	105
5225	5
*6985	400
7560	20
7600	205
10250	4050
15125	230
24500	1000

3.2- jadval Eng yaxshi moslashtirish algoritmi biroz murakkabroq, chunki maqsad vazifa uchun mos keladigan eng kichik xotira blokini topishdir:

Best-Fit Algoritmi

1. Xotira blokini (0) = 99999 ishga tushirish
2. Boshlang‘ich xotira yo‘qotilishi = xotira bloki (0) – vazifa hajmini hisoblash
3. Indeksni = 0 ga sozlash
4. Hisoblagichni 1 ga o‘rnatish
5. Hisoblagich < = xotiradagi bloklar soni bo‘lsa bajarish Agar vazifa hajmi > xotira hajmi (hisoblagich) bo‘lsa Unda, hisoblagich = hisoblagich + 1 Keyin Xotiradagi yo‘qotish = xotira hajmi (hisoblagich) – vazifa hajmi Agar boshlang‘ich xotira yo‘qotishlari > xotira yo‘qotishlari Unda, indeks = hisoblagich boshlang‘ich xotira yo‘qotishlari = xotira yo‘qotishlari hisoblagich = hisoblagich + 1 Bajarishni tugatish
6. Agar indeks = 0 bo‘lsa Unda, vazifani kutish navbatiga qo‘yish Keyin xotira maydoniga (pastki indeks) vazifani yuklash bo‘sh/band xotira ro‘yxatlarini o‘rnatish
7. Keyingi vazifaga o‘tish. Eng yaxshi moslangan algoritm bilan bog‘liq muammolardan biri shundaki, tanlovni amalga oshirishdan oldin, butun jadvalni qidirishi kerak, chunki xotira bloklari fizik xotirada joylashgan joyiga qarab ketma-ket saqlanadi (3.10-rasmda ko‘rsatilganidek, xotira qismlari hajmiga qarab emas). Tizim xotira qismining o‘lchamlari oshib boradigan tartibda ro‘yxatni doimiy ravishda tiklash algoritmini ishlab chiqishi mumkin, ammo bu qo‘shimcha xarajatlarni keltirib chiqaradi va uzoq muddatda qayta ishlash vaqtidan unumli foydalanishi mumkin emas. Eng yaxshi mos keladigan algoritm faqat bo‘sh xotira bloklari ro‘yxati bilan tasvirlangan. Jadvalda 200 ta bo‘sh joyni bloklash to‘g‘risidagi so‘rov endigina xotira menejeriga jo‘natildi. Eng yaxshi mos keladigan algoritmni qo‘llagan holda va ro‘yxatning yuqorisidan boshlab, xotira menejeri barcha ro‘yxatni qidiradi va 7600 manzilidan boshlab xotira blokini topadi, bu vazifani bajarish uchun yetarlicha katta bo‘lgan eng kichik blokdir. Xotira menejeri eng yaxshi mos keladigan algoritmdan foydalanib, ro‘yxatning yuqori qismidan boshlab, butun ro‘yxatni qidiradi va 7600 manzilidan boshlanadigan xotira blokini topadi, bu vazifani bajarish

uchun yetarlicha katta bo‘lgan eng kichik blokdir. Ushbu blokni tanlash bo‘sh joyni bexuda 110 sarflashni kamaytiradi (faqatgina 5K bo‘sh joy yo‘qotiladi, bu to‘rtta alternativ blokga qaraganda kamroq). Keyin vazifa 7600 manzildan boshlanib, keyingi 200 ta manzilni egallaydi. Eng yomon moslash usuli: ro‘yxatdan eng mos keladigan eng katta maydonni tanlashdir. Nima uchun eng katta? Bo‘linishning oldini olish uchun (bo‘linish muammosi ushbu ma’ruzada batafsil ko‘rib chiqiladi). Birinchi va ikkinchi strategiyalarni qo‘llash quyidagi nuqtai nazardan yaxshiroq: bajarilish tezligi va ishlatilgan xotiraning minimal hajmi bo‘yicha. Biroq, ulardan foydalanish bo‘linishni keltirib chiqarishi mumkin.

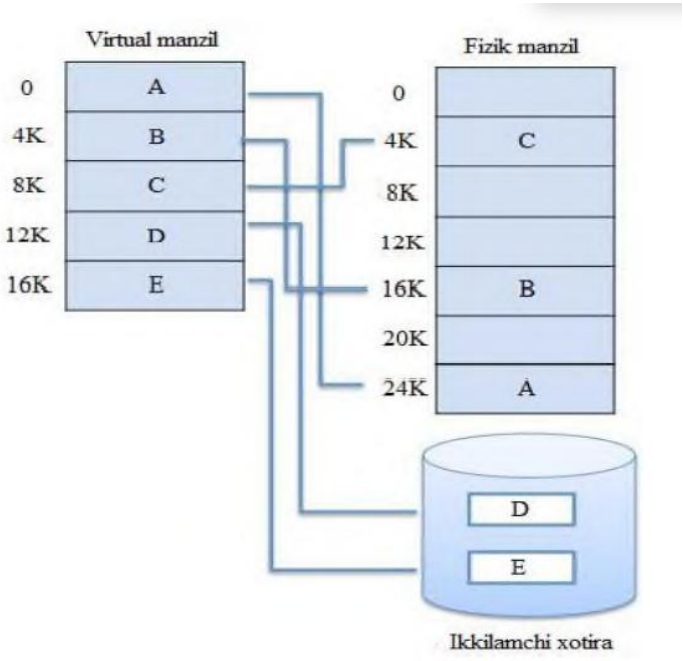
VIRTUAL XOTIRA

Virtual xotira kontseptsiyasi. Virtual xotirani sahifali tashkil etish. FIFO, LRU va algoritmlari

Virtual xotira (Virtual memory) Kompyuter tizimda o‘rnatilgan hajmdan ko‘proq xotiraga murojaat qilishi mumkin. Ushbu qo‘shimcha xotira aslida virtual xotira deb nomlanadi va u kompyuterning operativ xotirasini taqlid qilish uchun o‘rnatilgan qattiq diskning bir qismi hisoblanadi. Ushbu sxemaning asosiy ko‘zga tashlanadigan afzalligi shundaki, dasturlar fizik xotiradan kattaroq bo‘lishi mumkin. Virtual xotira ikki maqsadda xizmat qiladi. Birinchidan, bu bizga disk yordamida fizik xotiradan foydalanishni kengaytirishga imkon beradi. Ikkinchidan, bu bizga xotirani himoya qilishga imkon beradi, chunki har bir virtual manzil fizik manzilga tarjima qilinadi.

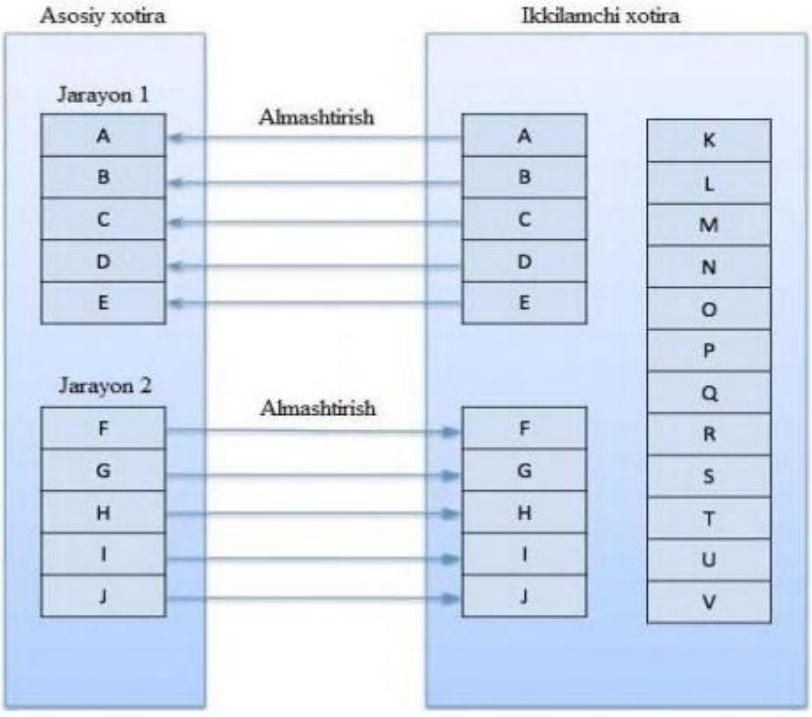
Dasturni asosiy xotiraga to‘liq yuklash talab qilinmaydigan holatlar quyidagicha:

- Foydalanuvchi tomonidan yozilgan xatolar bilan ishlash protseduralari ma’lumotlar yoki hisob-kitoblarda xatolik yuz bergan taqdirdagina qo‘llaniladi;
- Dasturning belgilangan parametrlari va funksiyalari kamdan-kam ishlatilishi mumkin;
- Aslida katta bo‘lmagan hajmdagi jadval amalda ishlatilgan bo‘lsa ham, ko‘p jadvallarga belgilangan miqdordagi manzil maydoni belgilanadi;
- Har bir foydalanuvchi dasturini xotiraga o‘rnatish yoki almashtirish uchun kamroq kiritish/chiqarish operatsiyalari talab qilinadi;
- Dastur endi mavjud fizik xotira miqdori bilan cheklanmaydi;
- Har bir foydalanuvchi dasturi kamroq fizik xotirani egallashi mumkin, shu bilan birga ko‘proq dasturlar bir vaqtning o‘zida ishga tushurilishi, protsessordan foydalanish va o‘tkazish qobiliyati mos ravishda oshadi.



3.22- rasm. Virtual xotira tizimi sxemasi

Umumiy foydalanish uchun mo‘ljallangan zamonaviy mikroprotsessorlar, xotirani boshqarish qurilmasi yoki xotirani boshqarish bloki (MMU), qurilma tarkibiga kiritilgan. Xotirani boshqarish blokining vazifasi virtual manzillarni fizik manzillarga tarjima qilishdir. Asosiy misol quyidagi 3.22- rasmda keltirilgan. Virtual xotira odatda talabni belgilash orqali amalga oshiriladi. U segmentlash yordamida ham amalga oshirilishi mumkin. Sahifani almashtirish (Demand paging) Sahifalarni almashtirish tizimi - almashtirish va sahifali tizimga juda o‘hshash. Jarayonlar ikkilamchi xotirada joylashadi va sahifalar oldindan emas, balki talabga ko‘ra yuklanadi. Kontekstni almashtirish sodir bo‘lganda, operatsion tizim eski dasturning biron bir sahifasini diskka yoki yangi dasturning biron bir sahifasini asosiy xotiraga ko‘chirmaydi. Buning o‘rniga u shunchaki birinchi sahifani yuklaganidan so‘ng yangi dasturni ishga tushiradi va ular bog‘langan dastur sahifalarini oladi.



3.23- rasm. Sahifani almashtirish tizimiga misol

Afzalliklari:

- Katta virtual xotira;
- Xotiradan yanada samaraliroq foydalanish;
- Ko‘p dasturlash darajasida chegara yo‘q;

Kamchiliklari:

- Sahifali boshqarishning oddiy usuliga qaraganda sahifalar soni va sahifani qayta ishlash uchun protsessor yuklanishi hajmi ko‘proq.

Sahifani almashtirish algoritmi. Sahifani almashtirish algoritmlari – bu usul operatsion tizim yordamida qaysi xotira sahifalarini o‘zgartirish, diskka yozish kerakligini hal qiladi. Sahifada xatolik yuz berganda va sahifani taqsimlash uchun ishlatib bo‘lmaydigan holatlarda, agar sahifalar mavjud bo‘lmasa yoki bo‘sh sahifalar soni talab qilinadigan miqdordan kam bo‘lsa, sahifani almashtirish har safar sodir bo‘ladi. O‘zgartirish uchun tanlangan va yuklanmagan sahifaga yana murojaat qilinganida, u diskdan ma’lumotlarni o‘qishi kerak va bu K/Ch tugallanishini talab qiladi. Ushbu jarayon sahifani almashtirish algoritmining sifatini aniqlaydi: sahifalarni joylashtirish uchun kutish vaqti qanchalik qisqa bo‘lsa, algoritm shunchalik yaxshi bo‘ladi. Sahifani almashtirish algoritmi qurilma tomonidan taqdim etilgan sahifalarga kirish to‘g‘risidagi cheklangan ma’lumotlarga qaraydi va sahifalarni o‘tkazib yuborishni kamaytirish uchun qaysi sahifalarni almashtirish kerakligini tanlashga harakat qiladi, uni dastlabki saqlash xarajatlari va algoritmning protsessor vaqti bilan taqqoslaydi. Sahifani almashtirish algoritmlari juda ko‘p. Bulardan eng ko‘p qo‘llaniladiganlari FIFO va LRU algoritmlari hisoblanadi.