Aja kvantimine - on vaja kindlalt paika panna, kui kaua mingi protsess saab protsessoriaega kasutada. Aja kvantimisel antakse igale protsessile protsessoriaega ajakvandi q (1..200 ms) jagu. Round Robini meetodi korral on katkestatud protsessid ja uued protsessid võrdväärsed. q tuleb õigesti valida - kui liiga väike, siis kulub mõttetult kaua palju aega ühelt protsessilt teisele ümberlülitumisel. Kui liiga suur, siis on mittesobivad ooteajad ja interaktiivse süsteemi puhul ei ole see hea. Üks võimalus on ka valida q sõltuvalt sisendjärjekorra pikkusest.

Alamprogramm - terviklik, iseseisev programmiosa, mida saab peaprogrammist või teisest alamprogrammist välja kutsuda korduvalt. Nende kasutamine muudab programmeerimise tunduvalt lihtsamaks ja ülevaatlikumaks.

Assotsiatiivmälu - mälu haldamise strateegia, kuis andmete poole ei pöörduta andmete aadressi, vaid sisu järgi. Nii on võimalik leida, kas mingi sõna leidub mingil mälu aadressil. Tavaliselt toimub otsing paralleelselt kõigi mälupesade suhtes. Võib ka otsida ligikaudset sarnasust (ntx. otsitakse, kus baidis teine bitt = 0 ja viies bitt = 1). Tavaliselt on assotsiatiivmälu väikese mahuga (tehnoloogilistel põhjustel paralleelse otsingu pärast). Tavaline näide - vahemälu.

Faili avamine - failide halduse alla kuuluv operatsioon, mis valmistab faili ette sealt lugemiseks või sinna kirjutamiseks. Hõlmab tavaliselt järgmisi tegevusi : leida etteantud kataloogist faili positsioon; kontrollida avaja õigusi (näiteks kas fail pole juba avatud, kaks protsessi korraga faili kirjutada ei saa); luua juhtblokk (FCB - File Control Block); luua identifikaator (kas FCB või muu); initsialiseerida faili marker.

Faili sulgemine - operatsioon, mille käigus antakse ära failile juurdepääsuõigus. Kontrollitakse identifikaatorit; kontrollitakse faili avatust; sünkroniseeritakse teiste protsessidega; eraldatakse mälu, kus vaja; parandatakse kataloog; tühjendatakse puhvrid (kirjutatakse faili andmed sisse); vabastatakse puhvrid; uuendatakse failimärgendid (lõpumarkerid ntx.); muudetakse infokandja olekut (ntx. lintmälu puhul keritakse linti).

Interpretaator - programm, mis teisendab arvutile võõra koodi (ntx. Basicu programmi) arvutile omasesse koodi. Teisendus käib lause-lauselt : lause teisendatakse ja täidetakse. Interpretaatori struktuur on võrreldes kompilaatoriga lihtsam, aga interpreteerimine on üldjuhul aeglasem kui kompileerimise kasutamine. Ntx. batch file.

Jadafail - fail, milles andmed on paigutatud jadamisi üksteise järele. Mingi kirje lugemiseks tuleb läbi lugeda kõik eelnevad kirjed, mistõttu otsimine on aeglane, aga järjest failist lugemine normaalne. (Ntx. lintandmekandjatel paiknevad failid on reeglina jadafailid). Andmete lisamisel faili lisatakse need kõige lõppu.

Järjekord - andmestruktuur, kus uued elemendid lisatakse lõppu ja elementide eemaldamisel võetakse need algusest (FIFO - First In-First Out). Sõnaga, kõige kauem oodanud element võetakse käsile kõige esimesena. Näiteks plaanurite juures kasutatakse järjekordasid - mingi taseme järjekorrast (valmis-, pooleliolevate või peatatud tööde järjekorrast) valitakse töö mingi kriteeriumi alusel (First-Come-First-Served, Shortest-Job-Next, Shortest Remaining Time Next, Round Robin, Event Driven).

Kaksikute meetod - mäluhaldamise mittelineaarne struktuur, kus vaba mälu vaadeldakse kahendpuuna. Ütleme, on mingi hulk mälu. Kui sellest tahetakse pool, siis jagatakse mälu kaheks ja antakse üks pool. Kui tahetakse veerand, siis jagatakse need kaks omakorda kaheks ja nii edasi kuni minimaalse mäluühikuni välja. Kui kaks kaksikut (ühest suuremast tükist tehtud kaks väiksemat) saavad vabaks, siis nad moodustavad jälle ühe suurema. Iga blokisuuruse kohta hoitakse omaette liste. Saab kiiresti ühe bloki aadressi põhjal leida tema kaksik ja vaadata, kas nad saab üheks suuremaks teha. Fragmenteerumise vältimiseks.

Kataloog - failide haldamise viis. Kataloog on loogiline struktuur, mis hõlmab mingite

parameetrite poolest sarnaseid faile. Ntx. köite kataloog (VTOC - Volume Table of Contents), süsteemi kataloog. Tavaliselt kataloog puukujuline, vaid viidete kogum failide füüsilistele aadressidele. Ntx. C:\DOS\COMMAND.COM - C-köites olevas kataloogis DOS asub fail COMMAND.COM. Arenenud süsteemides saab kataloogidele anda kasutusprivileegid, mis laienevad ka selles sisalduvatele failidele alamkataloogidele (ntx. RW, RO, NA). Kataloogi ülesanne on uuendada faili kohta käivat infot (nimi, asukoht, aadress, looja, omanik ja kasutajad, nende õigused).

Kirje - on informatsiooniväljade kogum, mida käsitletakse tervikuna.
Ntx.

```
type kirje = record
        eesnimi : string;
        perenimi : string;
end;
```

1. Võib olla fikseeritud pikkusega L. Sellisel juhul leiab kirje üles väga kähku, teades tema järjekorranumbrit k (pos = (k - 1) * L). Hea külg : väga lihtne andmeid muuta, sest ülejäänud kirjed ei nihku. Puudus : kui tervet pikkust L ei kasutata ära, läheb kasutamata ruum lihtsalt raisku.
2. Muutuva pikkusega. Kirje konkreetne pikkus paikneb kirje sees, nii et juurdepääs kirjele nõuab kirje skaneerimist. Eelis : mälu suhtes ökonoomne. Puudused : kirje pikkuse väljamaht võib probleeme tekitada, asendamine on tülikas.
3. Määramata pikkus. Kirjeid eraldavad tavaliselt kirje lõpus asuvad markerid (ntx. ";" - Pascalis, "." - tekstis, "0" - C-s, CR/LF tekstifailides). Füüsilisel tasemel tüüpiline kasutus. Eelis : ökonoomne mälukasutus. Puudused : juurdepääs kirjele eeldab kogu faili skaneerimist, tülikas asenduste tegemine.

Kommunikatsioon - protsesside vahel. Üks protsess saadab teate, teine võtab vastu. Teate formaat võib olla järgmine :

1. Saatja ID
2. Vastuvõtja ID
3. Pikkus
4. Tüüp
5. Keha

1-4 on fiks. pikkusega, 5 on fiks/var. Fiks. on mugav, odav. Var - paindlik, kallis. Otseadresseerimine - üks protsess saadab teate otseselt teisele protsessile. Parem süsteem on postkast. Vt. postkast.

Kompilaator - translaatorprogramm kõrgkeeles kirjutatud programmi teisendamiseks masinkeelde. Transleerimine võtab üldjuhul palju aega ja on mälunõudlik protsess. Täitmise ja transleerimise ühendatuse puhul on programmi raske osadeks transleerida; sellepärast tehakse üldjuhul transleerimise lõpus iseseisev ülesande kirjeldus e. objektmoodul. Et objektmooduli kood on suhtaadressides ja et puuduvad eraldi viidad alamprogrammidele, siis ei ole tavaliselt transleeritud programm e. objektmoodul valmis täitmiseks. Reeglina on transleerimise ja täitmise vahel veel hulk erinevaid funktsioone vaja realiseerida - põhimälu eraldamine ja jaotamine programmidele, süsteemsete alamprogrammide ja eraldi transleeritud programmide linkimine, üleminek suhtaadressidelt absoluutaadressidele, programmi koodi füüsiline salvestamine arvuti põhimällu. Neid funktsioone täidavad süsteemsed programmid : linkur ja laadur. Linkur ühendab enne laadimist eraldi asuvado bjektmoodulid terviklikuks täitmisprogrammiks ja salvestab saadud laademooduli üldjuhul välismällu. Laadur on programmiosa laademoodulite lugemiseks põhimällu.

Laademoodul - põhimällu lugemiseks ja seal täitmiseks sobivale kujule viidud programmiosa. Objektmoodulid ühendatakse linkuri poolt tervikuks ja salvestatakse reeglina välismällu.

Laadur - Vt. kompilaatori juttu lausest "Reeglina on..". Programm, mis laeb välismälust laademoodulid põhimällu. 3 liiki :
1. Absoluutne laadur - objektmoodul on absoluutaadressides, nii et programm laaditakse mälus

alati ühele positsioonile.
2. Häälestav laadur - objektmoodul on suhtaadressides. Laadida võib ükskõik millisesse mälupiirkonda, ülesandeks on järeltransleerimine. Enamikel juhtudel võib täitmise ümber laadida (saalida/swappida).
3. Linkiv laadur - ühendab enne laadimist eraldi asuvad objektmoodulid tervikuks ja salvestab mooduli üldjuhul välismällu.

Lehekülgmälu - mäluorganiseerimistehnika. Aadressi väli jagatakse fikseeritud pikkusega blokkideks (füüsilisteks lehekülgedeks). Andmed jaotatakse loogilisteks lehekülgedeks, mille suurus langeb füüsilistega kokku; füüsiline lehekülg laaditakse loogilisele leheküljele üks-ühele. Eelised - lihtne mälujaotus ja infovahetus. Puudus - linkimise keerukus (sellest võib üle saada segmentlehekülje organiseerimisega - sidemete organiseerimine programmi elementide vahel. Segmentlehekülje puuduseks on 3-etapiline pöördumine ülesande osa poole. Et vähendada nii tekkivat ajakulu, otsitakse tavaliselt paralleelselt mitut segmentlehekülge).

List - lineaarse struktuuriga andmekogum. Elemendid on seotud viitadega. Ntx. järjekorrad, maatriksid, pinud (mitte alati), puud (alati).

Monitor - protsesside sünkroniseerimiseks mõeldud seade, mis jälgib ja registreerib andmetöötlussüsteemis teatud toiminguid. Ntx. võib ühismuutujale olla ümber ehitatud monitor, mis hoolitseb, et ainult üks protsess korraga kasutab muutujat.

Mälu fragmenteerumine - töö alguses on põhimälu vähefragmenteerunud, aga töö käigus vabastatakse mälu ebaühtlaselt, nii et tekivad vabad augud. kuni lõpuks on põhimälu läbisegi täis vabu ja hõivatud blokke, nii et üht terviklikku andmekogumit peab hoidma eraldiasetsevates blokkides. See teeb andmekogumi lugemise aeglasemaks. (Eeltoodu ei käi ainult põhimälu kohta, aga kettamälu saab defragmenteerida vastavate programmidega). 50% reegel - vabade blokkide arv on pool hõivatud blokkide arvust (statistika põhjal). Võib kasutada mitmeid meetmeid fragmenteerumise takistamiseks (ntx. kaksikute meetod).

Objektmoodul - programmi on raske transleerida osadeks, kui täitmine ja transleerimine on ühendatud. Seetõttu transleerimise lõpus tehakse tavaliselt iseseisev ülesande kirjeldus e. objektmoodul. Üldjuhul ei ole objektmoodul ka valmis vahetult arvutis täitmiseks, kuna ta kood on suhtaadressides ja puuduvad eraldi viidad alamprogrammidele.

Ootamine - tekib, kui protsess ootab millegi järel (sisend/väljundandmete järel). Ooteaja lühendamiseks kasutada protsesside sünkroniseerimist ja tekitada listid *suspended* ja *ready* protsessidest. *Suspended* listis on need, kes mingit ressursi ootavad. *Ready* listis need, kes enam ei taha midagi muud peale protsessoriaja. Peab olema üles ehitatud katkestussüsteem / pidevalt üks jooksev protsess katkestatakse ja lastakse mingil teisel ready protsessil protsessorit kasutada.

Otsefail - erinevalt jadafailist võib otsefaili suvalisest kohast hakata lugema või kirjutama, teades andmete füüsilist asukohta osutavat aadressi. Ei pea olema mälus järjestikuliselt. Puudus - fragmenteerumine aeglustab faili lugemist. Tänapäeva arvutitehnoloogias on failid otsefailid, sest ei hoita lintmeedia peal üldjuhul.

Paiskadresseerimine - paigutusmeetod, mille juures on andmeid võimalik leida andmete väärtuse järgi. Andmete väärtusest arvutatakse välja ainult sellele väärtusele omane number ja see number ongi andmete paiknemise koht paisktabelis.

Pakett-töötlus - protsesside planeerimise viis, kus protsessid antakse protsessorile täita pakettide kaupa. Nii koormatakse protsessorit optimaalset. Pakett-töötlusesse saavad minna ainult protsessid, mis pole ajakriitilised ja võivad oodata (näiteks teksti sissetrükkimine ja kuvarile ilmumine ei tohi olla). Pakett-töötlussüsteemis on vajalik dünaamiline mälujuhtimine, paindlik protsessori ajajaotus, võimas käsukeel, juhtkeel ja peab olema võimaldatud operaatori osavõtt

riistvarajuhtimisest.

Pinu - andmestruktuur, mis on organiseeritud Last In-First Out (LIFO) magasinina. Viimasena lisatud element võetakse esimesena välja. Võib võrrelda ühe lahtise otsaga toruga, kuhu topitakse andmeid sisse üksteise peale. Mõnede arvuti mälustruktuuride juures kasutatakse pinu, näiteks jooksvate aadresside säilitamisel siirete ja katkestuste puhul.

Planeerimine - e. dispetsheerimine on süsteem, mille abil protsesside jagatakse protsessoriaega. Tavaliselt toimub kolmes osas :
1. kaugplaneerimine. Siin planeeritakse pakett-töötlust (valitakse protsesse, mis paketina täitmisele lähevad). Sobivad tööd suunatakse lähiplaneerimisele. Arvestab planeerimisel järgmisi kriteeriume : protsessori kasutus ja läbilaskevõime, käibeaeg, ooteaeg, reaktsiooniaeg, ülesande prioriteet, ülesande ressursinõuded.
2. Keskplaneerimine. Suunab peatatud tööd sekundaarmällu.
3. Lähiplaneerimine. Otsustab, millisele protsessile antakse protsessoriaega. Dispetšeri võivad käivitada järgmised sündmused : kell, katkestused ja I/O lõpp, interaktiivsed protsessid (klaviatuurisisend), *OS calls*, signaalid protsessilt; seda juhul, kui need sündmused muudavad globaalolekut. Võimalikud algoritmid :
1. First-Come-First-Served. Lühikesed tööd peavad taga ootama.
2. Shortest Job Next. Pikemad peavad ootama, aga inimmõistusele kõige tavalisem, ntx. poes, kes vähe ostab, võiks ette minna.
3. Shortest Remaining Time Next. Vajab infot aja suhtes. Ooteaja mõttes optimaalne, ohtlik pikematele töödele.
4. Round Robin. Vaata aja kvantimist.
5. Event Driven - teatud sündmuste toimumisel määratakse prioriteete.
6. Multiple Level Queue

```
System processes     --> High priority queue (ED)  ->
Interactive processes --> Medium priority (RR)     ->     CPU
Batch processes      --> Lowp riority (FCFS)       ->
```

Postkast - protsessorivaheline kommunikatsiooniviis. Üks protsess paneb postkasti teate, kust siis teised protsessid seda võivad kätte saada. On töökindlam kui otseadresseerimine, sest teine protsess ei pruugi alati olla valmis teate vastuvõtuks.

Prahikoristus - eraldatud mälu, mida enam ei vajata, ei vabastata kohe, vaid märgistatakse kui praht. Ja kui vaja (ntx. vaba mälu ei ole enam piisavalt), siis praht koristatakse - suunatakse vabade blokkide hulka. Eeliseks on see, et kui mälu on täis vaid hõivatud blokke ja prahti ja kui siis praht ära koristatakse, vabaneb mälu korraga suurte tükkidena, mis võimaldab mällu panna suuremaid tükke neid fragmenteerimata. Puuduseks on jällegi see, et kui mälu pole nii täis, et prahikoristust oleks vaja teha, aga on nii täis, et suuremaid vabu blokke enam pole, siis mälu hõivamisel fragmenteerub mälu jällegi rohkem.

Prioriteet - eelisõigus. Protsessidele omistatakse tavaliselt prioriteedid, mille järgi plaanurid/nile protsessoriaega jagavad. Näiteks sisend/väljundprotsessid on üldjuhul kõrgema prioriteediga, sest nad on ajakriitilised. Kui ma tahan, et arvutis muusika mängiks, siis ma tahan, et see toimuks viiveteta ja ühtlaselt.

Protsessi blokeerimine - blokeeritud protsess on *suspended* rezhiimis ja ootab mingit ressurssi. Ntx. ühismuutujat ei tohi kaks protsessi korraga muuta, siis kui üks protsess on ühismuutuja, on teisele protsessile ühismuutuja kasutamine blokeeritud.

Protsess - ettemääratud sündmusejada, mis leiab aset kogu programmi või mingi ta osa täitmisel. Ressursside suhtes võivad protsessid olla konkureerivad (pretendeerivad samale ressursile) või koopereeruvad (on seotud omavahel mingi ressursi kaudu, mida mõlemad muudavad. Ntx. klaver-klaveripuhver-kuvar). Reeglina võib protsess olla kolmes olekus : *suspended-ready-running*. Protsessi iseloomustavad veel tema nimi, prioriteet ja muu info ehk

veel.

Protsessi juhtimine - protsessi jälgimine ja kontrollimine. Juhitakse tavaliselt plaanurite abil. Vt. planeerimine.

Puhver - andmehoidla, mille kaudu saab ajutist salvestamist, kasutades vahetada andmeid erinevate edastusparameetritega objektide vahel. Puhvrid teevad üldjuhul töö märgatavalt kiiremaks. Näiteks kõvakettapuhvri puhul ei kirjutata andmeid pidevalt kettale, vaid puhvrisse, mis täitumisel, siis ühe hooga tühjaks kirjutatakse. Samuti ei pea protsessid ootama mingi aeglase väljundseadme taga, vaid annavad info puhvrisse, kust see võimaluse korral sellesse aeglasesse väljundseadmesse suunatakse.

Puu - mittelineaarne hierarhiline andmestruktuur. Algab tavaliselt juurest ja hakkab hargnema "oksteks" ja "oksadel" on "lehed" (andmed). Näiteks kataloogipuu. Puu võib olla ka kahendpuu - iga oks jaguneb maksimaalselt kaheks. Näiteks kaksikute meetodi korral kasutatakse kahendpuud. Annab enamasti ülevaatliku arusaadava pildi andmestruktuurist.

Register - sisemälu osa, millel on fikseeritud mälumaht ja mingi eriotstarve. Võimaldab säilitada ja taasesitada töödeldavaid andmeid või käske. Ntx. protsessori käsuregistris hoitakse käske.

Ressurss - andmetöötlussüsteemi iga element, mida operatsioonisüsteem võib kasutada. Igasugune mälu, sisend/väljundseadmed jne.

Seade - arvuti riistvarakomponent. Seadme tööd juhitakse tavaliselt operatsioonisüsteemi poolt seadmedraiveriga.

Segment - mälu on jaotatud segmentideks. Aadress on kaheosaline ja koosneb segmendi aadressist ja nihkest (info kaugus segmendi algusest). Edasi tuleb kahtlane jutt - segmenteerimine on programmi organiseerimisviis, kus programmi andmestruktuur peegeldab tema sisulist liigendust. Et segmente saaks ühendada tervikuks, peab segmentide tabelis iga segmendi kohta olam vähemalt neli tunnust :
1. T - ühekohaline, näitab, kus asub antud hetkel segment (0 - operatiivmälus, 1 - välismälus).
2. K - kaitsetunnus (kas Read-Only)
3. Segmendi baasaadress - kustmaalt mälus algab.
4. Pikkus. Iga programmi element on määratud segmendiga ja asukohga segmendis, seega on programmi poole pöördumine kaheetapiline.

Semafor - protsesside sünkroniseerimiseks kasutatav andmestruktuur, milelga reguleeritakse juuurdepääsu ressursile, mida ainult üks protsess võib korraga kasutada. Binaarne semafor (0,1). Ntx. kui semafor = 1, siis ressurss vaba. Kui protsess läheb ressurssi kasutama, siis semafor = 0 ja teised protsessid, mis ka tahavad muutujat kasutada, näevad, et semafor on kinni, ja jäävad ootama (järjekorda). Nüüd kui esimene protsess on lõpetanud, siis semafor = 1. Järjekorras järgmine protsess läheb muutuja kallale ja semafor = 0.
Võib olla ka numbriline semafor. Et üks protsess vähendab semafori, kuni see ei ole 0; kui on 0, siis ootab; ja teine protsess suurendab semafori, kui semafori taga parajasti keegi ei oota.

Sünkroniseerimine - kui protsessid on sõltumatud, ei pea neid sünkroniseerima. Kui aga protsessid kasutavad samu ressursse, siis peab. Näiteks peab sünkroniseerima protsesse, mis kirjutavad klaveripuhvrisse juurde (klahvivajutus) ja võtavad sealt ära (täht ilmus kuvarile). Peab olema kindlustatud vastastikune välistamine. Kas semafore või monitore või teadete süsteemi kasutades tuleb andmete kasutus reguleerida.

Teek - andmestruktuur. Evib sisemist nimekirja, kus on kirjas, mis elemendid seal paiknevad. Kahtlane jutt järgneb :
1. Tarbijale välismälus eraldatud piirkond, kuhu saab kanda programme, andmestikke ja muud infot.

2. Programmide ja alaprogrammide kogu, mida kasutaja saab programmis viidata ja kasutada.

Ummik - situatsioon, kus mingis protsessidekogumis kõik protsessid on pidevalt blokeeritud, oodates mingit ressurssi, mis ei tule (ntx. protsess, mis seda ressurssi annab, ootab ressurssi, mida saab anda üks ootav protsess) ja nii ei saagi ükski protsess aktiivseks. Et ummik tekiks, peab olema korraga täidetud 4 tingimust :
1. vastastikune välistamine (ressursi kallal 1 protsess korraga)
2. ootamine hoidmisega (muid ressursse ei vabastata)
3. puudub ressursside ümberjagamine
4. ringootus
Kui ühe tingimuse täitmine on takistatud, siis ummikut ei toimu. Ummikutega toimetulemise strateegiad on : ennetamine; välistamine; avastamine ja lõhkumine.
Ennetamine - ühe ummikutingimuse mahasurumine. Pole reaalne
Välistamine :
- kõik ressursid hõivatakse eelnevalt (kallis)
- inkrementaalne hõive - ootamisel vabastatakse muud ressursid (ohtlik, keeruline)
- mittekatkestuse tingimus - lubada katkestus (mõne ressursi puhul võimalik. CPU, mälu).
- ringootus - ressursside lineaarne järjestamine (klassidesse).
Kui hõivatud klass C_i, siis hõivata vaid C_{i+1}. Puudus - kindel järjestus.
Vältimine - kui ressursi hõivamine ei ähvarda ummikuga, siis eraldatakse.
Eeldusel, et iga protsess on oma ressursivajaduse deklareerinud, on kasutatav "pankuri algoritm" :

1. Ressursi tellmisel oletatakse, et see rahuldatakse.
2. Leitakse protsess i, mis saaks rahuldatud eksisteerivate ressursside eraldamisel.
    2a. olemasolevad ressursid + kasutatavad (i)
    2b. markeerida protsess i
    2c. korrata sammu 2
3. Kui kõik protsessid on markeeritud, on olek ohutu,

Vastastikune välistamine - kui muutujat tahavad muuta mitu protsessi, siis selle muutmine toimub kriitilises sektsioonis, kus samaaegselt võib olla ainult üks protsess. Vastastikuse välistamise mehhanism peaks kindlustama välistamise, mitte sõltuma süsteemist, mitte takistama ühisressursi mittekasutamisel konkurentsi, mitte eeldama midagi protsesside kiiruste, prioriteetide ja andmestruktuuride kohta. Mitmest ootavast protsessist üks peab pääsema kriitilisse sektsiooni lõpliku aja jooksul. Hea on kasutada semafore.

Virtuaalmasin - virtuaalne andmetöötlussüsteem, kus kasutajale emuleeritakse täiskomplektne arvuti oma operatsioonisüsteemiga ja mis näib olevat monopoolselt kasutaja käsutuses. Kasutaja käsutusse on antud teatud kettad (mis ei pruugi ühtida füüsiliste ketastega) ja teatud sisend/väljundseadmed. Põhimõtteliselt võib üks virtuaalmasin kasutada tervet olemasolevat mälu, see on interaktiivsetes rakendustes väga efektiivne. Interaktiivne kasutaja tahab saada kiiret reaktsiooni, aga reeglina mitte eriti suurt mahtu (teksti sissetoksimine ei nõua palju ressurssi).

Virtuaalmälu - mäluorganiseerimistehnika, mis annab programmi käsutusse suure ja pideva "põhimälu" bloki, mis tegelikkuses koosneb hoopis väiksemast blokist ja sekundaarmälu (kõvaketas vms) blokist. Nimelt on programmi täitmiseks piisav, et vahetult protsessoriga seotud mälus paikneb ainult täidetava programmi fragment, kõike muud võib hoida kusagil mujal. Eeliseks on suurenenud mälumaht ja süsteemi läbilaskevõime. Puudused on : ei toeta mittestruktuurseid programme; nõuab pidevat andmete edasi-tagasi liigutamist (saalimist), mis võtab aega - sekundaarmälu on reeglina tunduvalt aeglasem kui põhimälu. Siiski on ta äärmiselt kasulik süsteem.