DST203 4.c lekcija

Klasiskā procesora struktūra

Procesors ir datora centrālās vadības un informācijas apstrādes mezgls. Tā sastāvā ietilpst aritmētiski-loģiskais mezgls, vadība, universālo reģistru bloks īslaicīgai datu un adrešu uzglabāšanai un speciālie reģistri: komandu skaitītājs, komandas reģistrs, steka rādītājs un karodziņu reģistrs. Visus šos mezglus kopā saista universāla divvirzienu datu maģistrāle.
Procesors izpilda informācijas apstrādi saskaņā ar doto programmu un organizē datora centrālo vadību, nodrošinot visu datora agregātu saskaņotu darbību. Lai to nodrošinātu, centrālais procesors atšifrē un secīgi izpilda visas programmā norādītās komandas, vajadzīgā brīdī ierosinot citu sistēmas agregātu darbu, piemēram, griešanos pie operatīvās atmiņas vai ārējām iekārtām. Uzrādīto funkciju izpildīšanai izmanto šādas centrālā procesora sastāvdaļas.

1. Aritmētiski loģiskais mezgls veic datu un komandu aritmētisku un loģisku pārveidošanu. Tas parasti piemērots veselu skaitļu apstrādei, bet skaitļu ar peldošo komatu apstrādei lieto speciālu aritmētiski loģisko mezglu.
2. Vadības iekārta nodrošina kārtējās komandas saņemšanu no atmiņas, tās koda dešifrēšanu, operandu saņemšanu no atmiņas un komandā norādītās operācijas izpildīšanu aritmētiski loģiskajā mezglā, rezultāta ievietošanu atmiņā, kā arī procesora atsaukšanos uz kanāla (DMA) vai citu datora mezglu prasības pēc apkalpošanas.
3. Reģistru bloks noder par ātru vietējo atmiņu vadības informācijas uzglabāšanai. Tā adresējamos vispārējās nozīmes reģistrus pēc programmētāja vēlēšanās var izmantot par akumulatoriem, bāzes vai indeksu reģistriem, kā arī bieži lietojamo datu īslaicīgai uzglabāšanai.
4. Sakaru mezgls jeb interfeiss organizē informācijas apmaiņu ar operatīvo atmiņu vai ārējām iekārtām. Šodienas datoros tas vada standartizētu datora maģistrāli (agrāk komutāciju izpildīja ar multipleksoriem), kas nodrošina pārējo datora mezglu vienveidīgu sadarbību ar centrālo procesoru, kā arī dod iespēju daudziem ražotājiem vienlaicīgi izgatavot dažādus datora mezglus.
5. Dažādi palīgmezgli nodrošina pašreizējā laika uzskaiti, pārtraukumu pieņemšanu un apkalpošanu, sadarbību ar palīgprocesoru un DMA kontroleri, darbības traucējumu kontroli un diagnostiku, atmiņas daudzlīmeņu organizāciju, apkalpošanu un aizsardzību, un virkni citu funkciju.

Mikroskaitļotājos, kuros vispirms jānodrošina aparatūras ekonomija un pietiekami zema cena, visu bloku aparatūra ir ievērojami samazināta, un daudzu palīgmezglu vispār nav.
Tipisks procesora piemērs ir mikroprocesors i8080.
Procesorā komandas izpildīšana notiek laikā, un nepieciešams izšķirt vairākus stingri definētus laika intervālus:

- -katrs takts impulsu periods veido mazāko datora laika bāzi, kam atbilst datora stāvoklis un kurā tiek izstrādāti viens vai vairāki vadības signāli;
- -jebkura griešanās pie atmiņas vai ārējās iekārtas notiek pa datora maģistrāli un prasa maģistrāles ciklu, kas parasti ilgst vairākus takts periodus;
- -katras komandas izpildīšana prasa no atmiņas nolasīt vismaz operācijas kodu, bet bieži vien arī operandus, tāpēc komandas cikls sastāv no viena vai vairākiem maģistrāles cikliem.

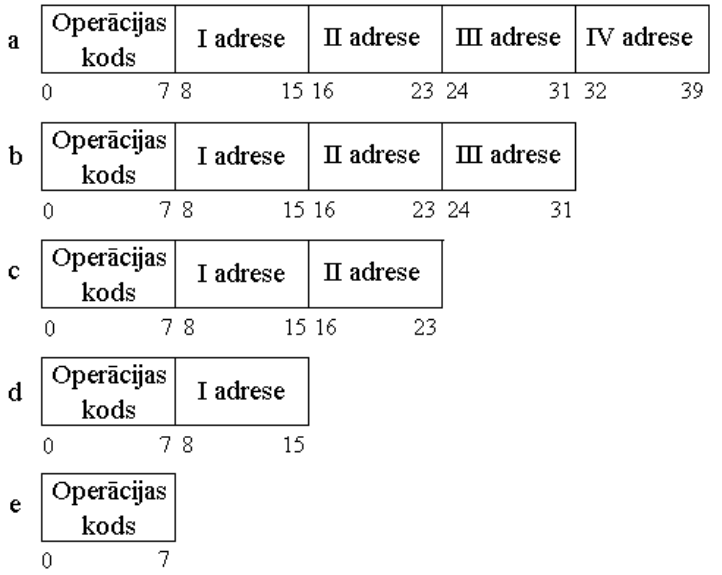
Šodienas datoros, lai palielinātu to ražību, lieto komandu un adrešu konveijerizāciju, vienlaicīgi apstrādājot vairākas komandas, kā arī virkni citu uzlabojumu.

Komandu struktūra

Katrai mašīnas komandai jānorāda,

1. kāda operācija mašīnai jāizpilda;
2. ar kādiem operandiem operācija jāveic;
3. kur jānovieto rezultāts;
4. kurai jābūt nākamai komandai.

Komandās parasti norādīti nevis paši operandi, bet gan adreses, kur šie operandi novietoti. Vispilnīgāk šī informācija parādīta tiešās adresācijas četradresu komandā (4.1. zīm. a), kurā norādītas abu operandu adreses (pirmā un otrā adrese), rezultāta adrese (III adrese) un nākamās komandas adrese (IV adrese).



4.1. zīm. Adresācija komandās.

a -četradresu; b -trīsadresu; c -divadresu; d -vienadresu; e -bezadresu.

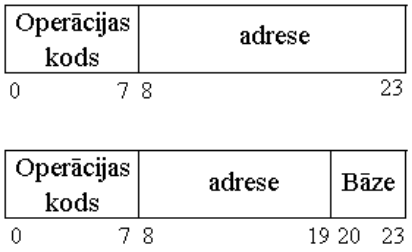
Tā kā programmā komandas galvenokārt seko pēc kārtas, tad nākamā izpildāmā komanda parasti atrodas atmiņas šūnā, kuras adrese ir par 1 lielāka. Tas ļauj no komandas izslēgt ceturto adresi un nākamās komandas adreses norādīšanai ievest komandu skaitītāju, kura saturs pieaug par 1 pēc katras komandas izpildīšanas. Veidojas 4.1. zīm. b parādītā trīsadrešu komanda.
Bet aprēķinu gaitā ļoti plaši par operandiem izmanto iepriekšējo darbību rezultātus, tādēļ trīsadrešu komandas stipri noslogo datu pārraides
ķēdes un atmiņu. Tāpēc lieto divadrešu komandu (4.1. zīm. c), kurā norāda tikai abu operandu adreses, bet rezultātu novieto vienā no šīm adresēm, parasti pirmajā.
Lielākā daļa moderno datoru, tajā skaitā arī visi PC tipa datori, izmanto modificētās divadrešu komandas, kurās visu atmiņas apjomu ļauj adresēt tikai viena adrese. Otra adrese norāda uz samērā nedaudzām superātrām operatīvās atmiņas šūnām -procesora iekšējiem reģistriem, kurās tiek novietoti visi starprezultāti.
Vienadreses komanda (4.1.zīm. d) norāda tikai viena operanda adresi, bet otrs operands tiek ņemts un rezultāts novietots speciāli šim nolūkam paredzētā reģistrā -akumulatorā. Vienadreses komanda uz datora ātrdarbības rēķina ļauj ievērojami samazināt izmantojamo aparatūru, tāpēc šādu komandu sistēmu plaši lieto mikroskaitļotājos.
Dažu komandu izpildīšanai nepieciešamā informācija jau atrodas noteiktos procesora reģistros, piemēram, akumulatora saturu bīdīt pa labi vai kreisi. Tādā gadījumā operanda adrese komandā vispār

netiek uzdota, un lieto bezadresu komandu (4.1. zīm. e).
Komandā gan operācijas kods, gan arī katra adrese dota ar divisko kodu, un skaitļotājā to norādīšanai tiek atvēlēti noteikti diviskā skaitļa biti.
Universālajos datoros komandas glabā tajā pašā atmiņā, kurā glabā skaitļus, un tās arī attēlojas kā diviskās sistēmas skaitļi. Tas nosaka divas būtiskas komandas īpašības:

1. komandas garumam jāatbilst datora informācijas reģistra garumam, un bez konteksta principā nevar atšķirt komandu no jebkuras citas informācijas;
2. ar komandām tāpat kā ar skaitļiem var izpildīt visas datorā paredzētās darbības.

Komandas adresācijas iespējas
Ikkatrs no operandiem un arī rezultāts var atrasties jebkurā atmiņas šūnā, tāpēc ciparu skaitam adresē jābūt pietiekamam jebkuras atmiņas šūnas numura norādīšanai. To nosaka bitu skaits adreses vārdā, kā arī bitu skaits maģistrālē, kas atļauj griešanos pie atmiņas. Nosacīti pieņemot komandas garumu 4 baiti jeb 32 biti un operācijas kodam atvēlot 8 bitus (4.1. zīm.), katram operandam iegūstam šādu tiešās adresācijas apjomu:

1. trīsadrešu sistēmā 8 bitus, kas ļauj adresēt 28=256 atmiņas šūnas;
2. divadrešu sistēmā 12 bitus, kas ļauj adresēt 212=4096 atmiņas šūnas;
3. vienadreses sistēmā 24 bitus, kas ļauj adresēt 224=16 777 216 atmiņas šūnas.



4.2. zīm. Adresācijas iespējas komandā.
a -ar tiešo adresāciju; b -ar bāzes adresāciju.

Tātad no atmiņas adresācijas viedokļa tieši uzrādot komandā adresi, visefektīvākā ir vienadresu sistēma.
Mikroskaitļotājos, kuru galvenais rādītājs ir to zemā cena, un kuros aparatūras ekonomijas nolūkā izmantots īss mašīnas vārds (8 vai 16 biti), iedarbīgs līdzeklis adresējamās atmiņas apjoma

paplašināšanai ir bāzes adreses izmantošana un ar to saistītā atmiņas lappušu organizācija. Šim nolūkam vienadresu komandas adrešu daļu sadala adreses un bāzes laukā (4.2. zīm.). Bāzes lauks norāda uz kādu no procesorā ievietotajiem reģistriem, un procesoram nepieciešami tikai daži biti šo reģistru numerācijai. Rezultātā adrese veidojas, komandas adrešu daļu saskaitot ar tā reģistra saturu, kura adreses kods atrodas bāzes laukā. Pie tam bāzes reģistra saturs parasti norāda vecākos adreses bitus jeb lappusi, bet komandas adrešu daļa -jaunākos adreses bitus jeb adresi lappusē. Tā, piemēram, 4.2.a zīm. aplūkotā komanda nodrošina 216=65 536 atmiņas šūnu adresāciju, bet 4.2.b zīm. redzamā komanda -16 bāzes reģistru un 212=4096 atmiņas šūnu adresāciju. Tā kā katrs bāzes reģistrs ir tikpat garš kā komanda un satur 24 bitus, tad kopējais tieši adresējamais atmiņas apjoms sastāda 212*224 = 236 6*1010 šūnas.
Komandas programmā parasti izpilda pēc kārtas, arī vairums ciklu noslēdzas komandas adrešu daļas iespēju robežās, tādēļ bāzes reģistru saturs jāmaina reti. Pat tad, ja programmas cikls atrodas divās blakus esošās lappusēs, cikla izpildīšanas laikā bāzes reģistru saturs nav jāmaina, tikai komandas pārmaiņus norādīs uz abiem bāzes reģistriem. Tas ļauj bāzes reģistru saturu izmainīt ar atsevišķu komandu tikai tā satura maiņas gadījumā un izpildāmās adreses numuru programmā norādīt saīsināti tikai ar komandas adrešu daļu. Rezultātā ievērojami samazinās komandas pieraksta garums un ar programmu aizņemtais atmiņas apjoms, kas kopā ar atmiņas lappušu organizāciju ļauj ievērojami palielināt datora ātrdarbību.

Adresācijas veidi
Moderno datoru komandas izmanto daudz un dažādus adresācijas veidus. Tos nosaka:

1. centrālā procesora struktūra;
2. atmiņas organizācija;
3. komandu formāti.

No tiem izplatītākie ir tiešā un netiešā adresācija, apslēptā, tūlītējā, bāzes, indeksējamā un relatīvā adresācija. Mikrokontroleros sakarā ar īso mašīnas vārdu bez tam plaši izplatīta ir adresācija ar rādītāju. Tā, piemēram, plaši izplatīto PC tipa datoru komandās izmanto 24 adresācijas veidus.
Tiešā (absolutā) adresācija attiecīgajos komandas laukos tieši norāda abu operandu un rezultāta adresi. Tā visskaidrāk izteikta 4.1b zīmējumā parādītajā trīsadrešu komandā.
Reģistru adresācija attiecīgajos komandas laukos tieši norāda izmantojamo procesora iekšējo reģistru.
Netiešā reģistru adresācijas gadījumā komandā norādītais reģistrs satur operanda adresi.
Netiešajā adresācijā komandas adreses daļa norāda tikai adresi, kurā atrodas meklējamā operanda adrese. Šī adrese savukārt var atkal norādīt adresi, utt. Lai netiešās adresācijas gadījumā atšķirtu adresi no operanda, komandas kodā ieved īpašu netiešās adresācijas pazīmi, kā tas raksturīgi datorā PDP11 un tā atvasinājumos. Šo adresācijas veidu plaši izmanto adrešu sarakstu apstrādei.
Apslēptās adresācijas tipisks piemērs ir 4.1c un 4.1d zīmējumā parādītās divadrešu un vienadreses komandas, kurās norādīta tikai divu vai tikai viena operanda adrese. Rezultāta un otra operanda adrese dota apslēptā formā kā noteikti aritmētiski loģiskā bloka reģistri, un šīs trūkstošas adreses komandas izpildīšanas gaitā tiek pieņemtas tā, kā tas paredzēts datora konstrukcijā.
Tūlītējā adresācija izmanto īsus operandus, un komandas adreses laukā uzrāda nevis operanda adresi, bet gan tieši tā vērtību. Šo adresācijas veidu plaši izmanto vadības maiņas komandās, kad komandas adreses laukā uzdots pieaugums, par kādu jāmainās komandu skaitītāja saturam, kā arī baitu apstrādes komandās.
Bāzes adresācijā izpildāmo adresi iegūst summēšanas ceļā, komandas adrešu daļu

saskaitot ar bāzes reģistra saturu. Šo paņēmienu plaši lieto mikroprocesoros, lai ar īsu komandas vārdu adresētu lielus atmiņas apjomus.
Relatīvajā adresācijā izmanto īpašību, ka par bāzi var noderēt jebkurš datora reģistrs, tāpēc par bāzi izmanto komandu skaitītāju, ar komandas adrešu daļu norādot nobīdi attiecībā pret komandu skaitītāju.
Pēdējais paņēmiens sevišķi ērts vadības nodošanas komandu veidošanai, ar to iegūstot programmas, kuras bez pārtranslēšanas var ievietot jebkurā atmiņas vietā.
Indeksējamā adresācijā, tāpat kā bāzes adresācijā, izpildāmo adresi veido ar summēšanu, saskaitot komandas adrešu daļu ar indeksu reģistra saturu. Būtiska atšķirība šeit ir tā, ka indeksu reģistrs satur adreses jaunākos bitus tāpat kā komandas adrešu daļa. Pēc katras komandas izpildīšanas indeksu reģistram pieskaitot vai atskaitot 1, iegūst ***autoinkrementovai autodekremento adresāciju***, kas sevišķi ērta informācijas masīvu apstrādei, kad operands dots kā masīva elements ar savu kārtas numuru.
Bāzes indeksējamā adresācija tiek bieži pielietota masīvu apstrādei, ar bāzes adresi nosakot masīva sākuma adresi atmiņā, bet indeksu norāda masīva elementa kārtas numurs (piemēram, PC datoros).
Adresācija ar rādītāju ir netiešās adresācijas paveids, kura sakarā ar īso komandas vārdu tās adrešu lauki doti apslēptā formā vai arī ir ļoti īsi (3 vai 4 biti) un norāda uz vienu no reģistriem procesora iekšējo reģistru blokā, kurš satur operanda adresi (piemēram, HL reģistrs i8080 procesorā).
Steka adresācija ir adresācijas ar rādītāju paveids, kurā par rādītāju izmanto steka rādītāju.
Vadības nodošanas komandas var izmantot visus adresācijas veidus, un šo komandu izpildes gaitā nākamās komandas adrese veidojās, iegūto operandu iesūtot komandu skaitītājā. Tāpēc nākamā komanda var atrasties jebkurā atmiņas vietā. Jāatgādina, ka normālas komandu izpildes secības gadījumā nākamās komandas adrese ir par vienu lielāka par iepriekšējo.

Adresācija dažādās sistēmās
Kā jau norādīts, atkarībā no procesora un atmiņas organizācijas katram datoru tipam ir savas adresācijas iespējas.

PDP11.

Miniskaitļotāju PDP11 komandu sistēma satur komandas darbībām kā ar vienu, tā divu baitu gariem vārdiem, izmantojot vienadresu un divadresu komandas, ka arī atmiņas adresēs iekļauto ārējo iekārtu adresāciju. Tiešā un netiešā reģistru, autoinkrementā, autodekrementā un indeksējamā adresācija kopā ar steka atmiņas organizāciju un mainīgu komandu formātu nodrošina samērā lielās skaitļošanas iespējas.
Komandas pēc savas uzbūves sadalās:

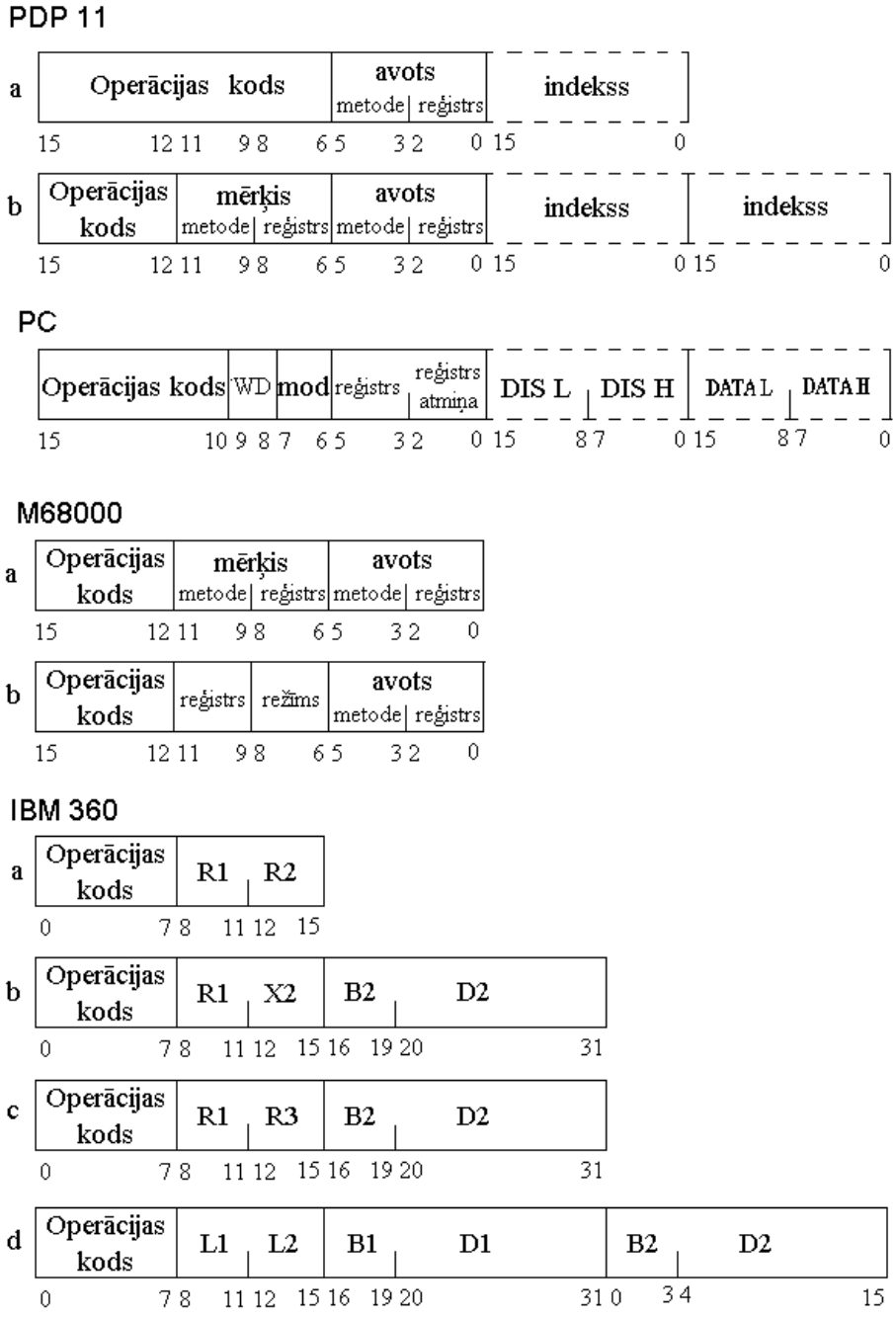
- bezadresu komandās, kuras satur tikai operācijas kodu;
- vienadresu un divadresu komandās, kuras satur operācijas kodu, vispārējās nozīmes reģistra numuru un adresācijas metodes norādījumu.

Izvēlētā komandu sistēma speciāli paredzēta regulāru datu struktūru apstrādāšanai. Vispārējās nozīmes reģistri šajās procedūrās tiek izmantoti:

- datu glabāšanai;
- kā adresu rādītāji;

- kā adresu rādītāji, kuru saturs mainās automātiski ar uzdoto soli;
- kā indeksu reģistri, kuru saturs operanda adreses izskaitļošanas gaitā saskaitās ar komandā uzdoto indeksa vārdu.

Datu apstrādes komandas izmanto vienadresu un divadresu komandu formātus. Vienadresu komandas formāts dots zīm. 4.3. PDP11a, kur komandas biti 0-5 atkarībā no izpildāmās darbības norāda avota vai uztvērēja lauku, atsevišķi norādot kā izmantojamā vispārējās nozīmes reģistra numuru (biti 0-2), tā arī izmantojamās adresācijas veidu (biti 3-5). Biti 6-15 norāda izpildāmās operācijas kodu, pie kam 15 bits norāda uz operācijas izpildīšanu ar baitu (1) vai vārdu (0).
Divadresu komandas formātā (zīm. 4.3. PDP11b) uzdoti kā uztvērēja (biti 0-5), tā avota (biti 6-11) lauki, operācijas kodam atvēlot 12-15 bitus. Katrs lauks programmas pierakstā reprezentējās ar vienu astotnieku sistēmas ciparu.



4.3. zīm. Adresācija dažādās sistēmās

Vadības nodošanas komandu formāts ir sarežģītāks, un to laukos papildus norādīts gan vispārējās nozīmes reģistrs, kurš piedalās komandas izpildīšanā, gan arī nobīde par 8 (xxx) vai 6 (NN) bitiem attiecībā pret komandu skaitītāju.
Tiešās adresācijas metodes piemērotas datu masīvu apstrādei, un attiecīgās metodes kods kā astotnieku sistēmas cipars ievietojās komandas metodes laukā. Divadresu komandās katrs no operandiem var būt uzdots ar jebkuru adresācijas metodi.
Netiešās adresācijs metodes ērtas gadījumā, ja informācijas tabulas satur operandu adreses, nevis pašus operandus.
Izmantojot komandu skaitītāju par vispārējās nozīmes reģistru, jāievēro, ka katra griešanās procedūra pie atmiņas ar komandu
skaitītāja izmantošanu izsauc tā satura palielināšanos par 2, kas
vietā arī vienbaita operācijās.
Komandu skaitītāju var izmantot visās adresācijas metodēs, bet vislielākā nozīme ir tiešās, absolūtās, relatīvās un netiešās relatīvās adresācijas metodēm. Šīs metodes atļauj veidot programmas, kuras izpildās neatkarīgi no to novietošanas vietas atmiņā.

Intel 80x86 un Pentium.
Visa IBM PC saimes komandu sistēma balstīta uz i8086 komandu sistēmas bāzes. Procesoru sarežģītā iekšējā struktūra, atmiņas segmentu organizācija un daudzās adresācijas metodes prasa pielietot dažādus komandu formātus, kuri apvienoti 113 pamata tipos, nodrošina 24 adresācijas veidus un ļauj izpildīt sekojošas pamatdarbības: datu pārsūtīšanu, aritmētiskās un loģiskās darbības, datu ķēžu apstrādāšanu, vadības nodošanu un pārtraukumu apstrādāšanu. Procesora arhitektūra paredzēta vārdu apmaiņai ar atmiņu, par otru operandu izmantojot tā iekšējo reģistru saturu. Procesora komandas ir 1 -6 baiti garas, un tipiskais divadresu komandas formāts dots 4.3. zīmējumā.
Komandas kods aizņem pirmā vārda 10-to līdz 15-to bitu. Komandas 9-tais bits satur rādītāju, kurš norāda uz vārda (W=1) vai baita (W=0) apstrādāšanu. Virziena rādītājs D norāda, vai darbības rezultāts tiks nosūtīts uz bitos 3 -5 norādīto reģistru (D=1), vai uz laukos mod un reģistrs/atmiņa norādīto adresi (D=0).
Lauks mod specificē lauka reģistrs/atmiņa izmantošanas veidu, norādot, vai operands atradīsies procesora iekšējā reģistrā (mod=11), vai adrese būs jāizskaitļo operatīvajā atmiņā (2 -4 saskaitāmie). Otrs operands vienmēr atrodas procesora iekšējā reģistrā, kura adresi norāda komandas 3 -5 bits (lauks reģistri).
Vajadzības gadījumā komandas pamata laukam pievieno baitus DIS L un DIS H, kuri koriģē reģistru saskaitīšanā iegūto adresi, kā arī tiešo operandu saturošos baitus DATA L un DATA H.

Motorola
Šīs sistēmas mikroprocesoru komandu sistēma ar dažiem izņēmumiem analoga PDP11.

IBM 360
IBM 360 un padomju EC tipa datori ir ar atmiņas baitu organizāciju, bet to aritmētiski -loģiskā mezgla reģistri operē ar 4 baiti gariem vārdiem. Procesorā ievietoti programētājam pieejami 16 vispārējās nozīmes reģistri, kuru adresācijai nepieciešami tikai 4 biti, kas ļauj samazināt komandas garumu un tās aizņemto vietu atmiņā. Tā rezultātā, operācijas kodam atvēlot 8 bitus, divadresu komandas darbam ar operandiem, kuri atrodas procesora iekšējos reģistros, ir tikai divi baiti garas (4.3. zīm. IBM a).
Procesora iekšējā reģistra 24 jaunākos bitus izmantojot bāzes adreses B un indeksa X glabāšanai, iespējams adresēt 224 16*106 atmiņas šūnas. Katrā komandā uzdotā nobīde ļauj adresēt vienu 212 = 4 Kbaiti garu lappusi (4.3. zīm. IBM b). Komandā paredzēta arī iespēja izmantot grupveida operācijas (4.3. zīm. IBM c), kur viens operands atrodas reģistros R1 līdz R3 ieskaitot, bet otrs operands atmiņā ar adresi A = (B2) +D2. Daudzbaitīgās aritmētikas vajadzībām paredzēta komanda ar 4.3. zīm. IBM d

parādīto formātu, kurā lauki L1 un L2 norāda operandu garumu 1 -16 baiti, bet operandu jaunākā baita adresi uzdod bāzes reģistra B satura un nobīdes D summa.