Számítógép architektúrák kidolgozott tételsor

Verzió: 1.0 (Build:1.0.2010.12.31.)

Készült:

- Számítógép architektúrák elméleti (Dr. Fazekas Gábor), gyakorlati (Pánovics János) jegyzetek
- Operációs rendszerek I. tananyag
- Kocsis Ilona és Győri László közös jegyzete
- Dr. Csernoch Mária Bevezetés az informatikába Power Point dokumentumok
- Ajánlott irodalmak
- Intel® 64 and IA-32 Architectures Software Developer's Manual Volume 1: Basic Architecture
- Hasonló témájú dokumentumok, jegyzetek
- Csoporttársak javaslatai
- valamint egyéb felsőoktatási intézmények oldalán talált dokumentumok alapján.

Megjegyzés:

- A 10-es tétel nem biztos, hogy 100%-osan jó

Készítette: Boruzs Tibor

(Figyelem! Bukásért felelősséget nem vállalok!)

2010/11-es tanév, 1. félév

1. A gépi adatábrázolás kategóriái

A számítógép a bemenő adatokhoz kimenő adatot rendel. A Bemenő adatot a felhasználó irányítja. Az adat a jelentésétől megfosztott információ.
Adatábrázolás:

Adat:

- logikai (Pl. Barkóba kérdéssorozatra válasz)
- numerikus
    - decimális, bináris, hexadecimális
        - fixpontos, lebegőpontos

A számítógép kettes számrendszerben működik. Az információ alapegysége így a bináris számjegy, a bit lesz, ezek pedig a 0 és az 1. Bináris számok ábrázolásakor ugyanúgy helyi értékes felírást használunk, mint a decimális számok esetén. Pl. a 10011101 bináris számnak 128 + 16 + 8 + 4 + 1 = 157 az értéke. Az egyes helyi értékek jobbról-balra 2-nek egymás után következő hatványai, tehát 1,2,4,8,16 stb.. Egy aránylag kicsi szám bináris leírásához sok 0-t és 1-et kell leírnunk, ez pedig néha fárasztó. Ezt kiküszöbölendő a számokat sokszor írjuk tizenhatos (hexadecimális) számrendszerben. Itt a számok a 10 arab számjegy, és az angol (latin) ábécé első hat betűje (A,B,C,D,E,F), A = 10,B = 11 stb. Mivel $16 = 2^4$, ezért négy bináris számjegy éppen egy hexadecimális számjegyet tesz ki. Az előző példa alapján 10011101b = 9Dh = 157d. Most elevenítsük fel a legegyszerűbb, közismert logikai műveleteket. A két logikai igazságértéket itt most bináris számjegyek fogják jelölni: 1 jelenti az „igaz" értéket, 0 a „hamis" értéket. A negáció (tagadás–negation) egyváltozós művelet, eredménye a bemeneti igazságérték ellentettje, jelölje NOT az angol tagadás mintájára. A konjunkció („ÉS") már kétváltozós (bináris)művelet. Jele AND (az „és" angolul). A diszjunkció („VAGY") szintén binárisművelet. Jele OR (a „vagy" angolul), és a két változón MEGENGEDŐ VAGY műveletet hajt végre. Az antivalencia („KIZÁRÓ VAGY", az ekvivalencia tagadása). Jele az XOR (eXclusiveOR).

2. Fixpontos számábrázolási módok: előjeles-abszolútértékes, többletes számábrázolás.

Előjeles, abszolút értékes:

- előjel bit
    - legmagasabb helyi értéken (balról az első bit)
        - 0: pozitív; 1: negatív
    - maradék bitek a szám ábrázolására, bináris, a szám abszolút értéke
- jellemzők
    - a nulla kétféleképpen ábrázolható
    - a legkisebb szám: −127
    - a legnagyobb szám: +127

Többletes:

- a szám és a többlet összegét ábrázoljuk binárisan
    - pozitív szám
    - m bites szám esetén a többlet
        - legmagasabb helyi értéken 1, a többi 0,
        - legmagasabb helyi értéken 0, a többi 1
- jellemzők (128 többlet esetén)
    - a nulla egyértelműen ábrázolható
    - a legnagyobb szám: +127
    - legkisebb szám: −128

3. <u>Fixpontos számábrázolási módok: kettes komplemens ábrázolás</u>

- előjel bit
  - legmagasabb helyi értéken (balról az első bit)
    - 0: +
    - 1: –
- maradék bitek a szám ábrázolására
  - bináris
  - pozitív szám
    - szám
  - negatív szám
    - 1-es komplemens
    - 1-es komplemens +1
- jellemzők
  - a nulla egyértelműen ábrázolható (pozitív)
  - a legkisebb szám: –128
  - a legnagyobb szám: +127

4. <u>Aritmetikai műveletek (összeadás, kivonás, negatív képzés) a kettes komplemens formátumban ábrázolt számok között, a túlcsordulás detektálása.</u>

Egy bináris szám kettes komplemensét úgy képezzük, hogy a szám inverzéhez hozzáadunk 1-et.
Példa a kettes komplemens képzésére: n = 8, N = -9
N = 9 00001001B
inverze 11110110B
<u>+1 00000001B</u>
komplemens kódban: N = -9 11110111B
Egy 8 bites regiszterben kettes komplemens kódban ábrázolható számtartomány határai:
pozitív számok: 00000000B = 0 ..... 01111111B = 127
negatív számok: 11111111B = -1 ..... 10000000B = -128

Összeadás és kivonás komplemens kódban

A továbbiakban csak az összeadás problémáit fogjuk megvizsgálni, hiszen a kivonás komplemens kódban összeadássá alakul: a kisebbítendőhöz hozzáadjuk a kivonandó kettes komplemensét (-1 szeresét).

Két negatív szám összege mindig negatív:
X = -2 1110B
<u>Y = -3 1101B</u>
11011B Az átvitelt elhagyva az eredmény helyes (-5). Komplemens kódú összeadáskor, ha mindkét operandusunk negatív, mindig *keletkezik* átvitel, különböző előjelű operandusok esetén *keletkezhet*, pozitív operandusok esetén *nem keletkezik* átvitel**. Az átvitelt minden esetben figyelmen kívül kell hagyni.** Nézzük meg azt az esetet is, ahol az eredmény nem fér el 3 biten.
X = 4 0100B
<u>Y = 5 0101B</u>
1001B
A két **pozitív** operandus összege **negatív** lett! Nem fértünk el az ábrázolási tartományban, **túlcsordulás** (overflow) keletkezett.
X = -7 1001B
<u>Y = -5 1011B</u>
0100B
A két **negatív** operandus összege **pozitív** lett! Ismét **túlcsordulás** keletkezett, mert a -12 komplemens kódban 4 biten nem ábrázolható.

5. <u>Kettes komplemens formátumú számok szorzása. Booth algoritmus alapötlete.</u>

Az algoritmus a nem inverz (egyenes) kódhoz hasonló, a különbség csupán annyi, hogy amikor a szorzó legmagasabb helyi értékű bitjével szorzunk, akkor a részletszorzatot nem hozzáadjuk, hanem levonjuk a részletösszegből. A kettes komplemens kódú ábrázolás úgy is tekinthető, mintha a

legmagasabb helyértékű bit súlyozása negatív, a többié pedig pozitív lenne. Mivel azonban a szorzandó maga is lehet, hogy negatív szám, a részletösszeg jobbra léptetésekor az előjelbitre különös tekintettel kell lennünk, azaz a korábbi előjelbitet jobbra léptetjük, ugyanakkor a legmagasabb pozícióban változatlanul megismételjük ("csíkot húz maga után"). Azért lassú egy szorzás, mert sok 1-es van a szorzó bitjeiben s ez sok összeadást jelent. Az algoritmus próbálja leegyszerűsíteni úgy, hogy kevesebb egyes legyen benne. Nem működik minden számra. Azoknál jó, ahol hosszú, csupa egyes vagy csupa nullás csoportok vannak.
A konkrét algoritmus helyett az elv: Például 62-vel kell szoroznunk.

0011|1110 = 62

Mivel 5 egyest tartalmaz, ez 5 egységnyi műveletet eredményez (az összeadást tekintve egységnek)

Gyorsítás: 62 közel van a 64-hez 64-2= 2^6-2^1

Így a szorzandót megszorzom 64-gyel, majd kettővel és veszem a két eredmény különbségét:

0100|0000	64
-0000|0010	-2

A binárisan végzendő műveletek:

- hatszoros léptetés balra;
- a egyszeresének hozzáadás a gyűjtőhöz;
- egyszeres léptetés balra;
- a szorzandó egyszeresének hozzáadás a gyűjtőhöz;
- a két eredmény kivonása egymásból.

Mivel a léptetés rendkívül gyors, tehát összesen 2 db összeadást, majd 1 db kivonást végzek, ami 3 db egységnyi műveletet jelent. Az eredeti megoldással szemben **40%-os gyorsítást sikerült elérnünk!**

6. **Lebegőpontos számábrázolás: IBM és IEEE-754 szabványok.**

IEEE 754:

- előjel bit (1 bit: 0->+, 1-> -)
- karakterisztika: hatványkitevő (8 bit)
- mantissza: szám normalizált értéke (23 bit)
- normalizálás
  - egészre normalizálás
  - törtre normalizálás
- bináris számrendszerben normalizált
- egészre normalizált
- karakterisztika: 127 többletes
- előjel
  - pozitív szám: 0
  - negatív szám: 1

IBM szabvány:

Hasonló, csak a normalizálás nem 2 hatványaival, hanem 16 hatványaival történik

- előjel bit
- karakterisztika 7 bit, 64 többletes
- mantissza 24 bit

7. **A Neumann-elvű számítógép fő funkcionális egységei: központi egység, I/O-vezérlő egység(ek), perifériák.**

Központi egység

Részei:

- Központi feldolgozó egység (CPU = mikroprocesszor)
  - Vezérlőegység (CU):

A vezérlőegység (angolul: Control Unit) szerepe a számítógép működésének, tehát a műveletek program szerinti végrehajtásának az irányítása is. Ez az egység rendszerint a következő részegységekből épül fel:

- programszámláló (IP)
- veremtár- (memória-cím) mutató (Stack Pointer)
- Utasításdekódoló
- Órajel generátor

Be- és kiviteli vezérlő egység (Input/Output-vezérlő)
A számítógép és az ember, valamint a számítógép és az általa vezérelt berendezések közti kapcsolatot a ki/beviteli egység teszi lehetővé (Input/Output Unit). Általában a ki/beviteli egység felelős a számítógép és a külvilág közti információcseréért. A beviteli egység segítségével visszük be a számítógép operatív memóriájába a megoldandó feladatot leíró programot és az ehhez szükséges adatokat. A számítógép ki/beviteli egységéhez kapcsolt ki-, illetve beviteli készülékeket (például: lyukszalag olvasó, billentyűzet, egér, nyomtató, katódsugárcsöves megjelenítő) perifériáknak nevezik. A perifériák egy sajátos illesztőegységen, interfészen keresztül csatolódnak a ki/beviteli egységhez.
Perifériák
Feladatuk: az adatok be- és kivitelét valósítják meg
Típusaik:

- Egyfunkciós perifériák (az adatáramlás csak egyirányú lehet, pl: billentyűzet, monitor)
- Párbeszédes vagy kétfunkciós perifériák (az adatáramlás kétirányú lehet pl.: terminál)
- Háromfunkciós perifériák (háttértárak)

8. <u>A központi egység fő funkcionális elemei (CPU, operatív tár, busz-rendszer).</u>

CPU processzor, amely feladata a számítógép vezérlése (CU) és az aritmetikai logikai műveletek (ALU) elvégzése. A processzorokat a műveleti sebességgel (MIPS), órajel frekvenciával (GHz) jellemzik.
Az operatív tár központi vezérlőegység által közvetlenül címezhető, elérhető tárolóegység, azaz memória. Tartalmazza a végrehajtás alatt álló program utasításait, valamint adatait. A CPU önmagában nem elegendő ahhoz, hogy sikeresen végrehajthassa a feladatát: az adatfeldolgozáshoz olyan tároló hely is szükséges, ahol a számítógép ezekhez gyorsan hozzá is férhet. Azokat az adatokat, amelyek tehát egy adott feladat megoldásához szükségesek, tárolni kell, de úgy és ott, ahol a CPU könnyen és gyorsan hozzájuk tud férni. Ez a hely a memória, vagy más néven az operatív tár.
A számítógépekben az egyes ki- és beviteli egységek közötti adatáramlás elektromos vezetőkön keresztül valósul meg. A jeltovábbító vezetékeket csoportokba foglalják, és ezeket nevezik síneknek (az eredeti angol: bus, azonban a magyar nyelvben nem busznak, hanem sínnek nevezzük).

9. <u>A CPU felépítése és főbb funkcionális elemei: vezérlő egység, aritmetikai-logikai egység (ALU), lebegőpontos egység (FPU), memória kezelő egység (MMU), regisztertár, belső busz, gyorsító tár(ak) (cache). Órajel-generátor.</u>

A vezérlőegység (angolul: Control Unit) szerepe a számítógép működésének, tehát a műveletek program szerinti végrehajtásának az irányítása is. Számítógépben értelmezi a tárolt program utasításait és biztosítja az utasítások helyes sorrendben történő végrehajtását. A vezérlőegység működésmódját a számítógép utasításrendszere, valamint utasításkészlete határozza meg.
Az aritmetikai és logikai egység (ALU – Arithmetic and Logic Unit), amint az elnevezése is mutatja, azon aritmetikai és logikai műveletek végrehajtását teszi lehetővé, amelyekkel a program által meghatározott számolási és logikai műveletek sorozata végezhető el.
Lebegőpontos egység (fpu): A számítógépek lebegőpontos (nem egész számokkal dolgozó) számításokat végző feldolgozó egysége. Az FPU a modern processzorokban a CPU-val egy lapkára

kerül integrálásra, de korábban megszokott volt, hogy külön tokban kapott helyett, és opcionálisan lehetett megvásárolni matematikai KO PROCESSZOR formájában.
A központi egység fel van szerelve egy úgynevezett memóriakezelő egységgel (MMU), amely figyeli, hogy olyan kódrészre kerül-e a vezérlés, amely nincs benn a központi memóriában (mert például a háttértárra van kirakva). Ha a memóriakezelő egység azt találja, hogy ez az eset áll fenn, akkor az operációs rendszert arra utasítja, hogy rakja ki a háttértárra a folyamatnak azt a részét, amely jelenleg a memóriában van, és azt a részt hozza be a helyére, amelyre ezután szükség lesz.
Regisztertár: A CPU rendelkezik egy belső regisztertömbbel. A regiszterek a CPU belső tároló elemei. Tartalmuk gyorsan és egyszerűen elérhetők a CPU elemei számára. "Munkamemóriát" biztosítanak az ALU számára, ideiglenes tárolást biztosítanak, segítik a címképzést, tárolnak állapotjellemzőket, státusokat, ezzel a vezérlést segítik. Különböző hosszúságúak (1-2 byte-os, szó stb.).

- A regiszterek általában nagyobb egységet alkotnak, melyeket regisztertáraknak nevezünk
- Általában a regisztertár csak egy részét képes egy-egy folyamat elérni. Ezek a regiszterek tulajdonképpen a folyamat szempontjából egy ablaknak tekinthetők.

A belső busz kapcsolatot tart a CPU alkotórészei között (alu, fpu, stb. közt)
Cache: Célja az információ-hozzáférés gyorsítása. A processzorok mindig gyorsabbak a memóriáknál (akár 5x-20x). A CPU lapkára integrálható memória gyors, de kicsi. Amikor egy utasításnak adatra van szüksége, akkor először itt keresi, ha nincs itt, akkor a központi memóriában.
Az órajelgenerátor (Clock Generator, Timing Unit) állítja elő a gép időbeni működéséhez szükséges vezérlőjeleket. Ezek rendeltetései a következők: az aritmetikai és logikai egység vezérlése (az utasításdekódoló által jelzett művelet elvégzése), az információk kiolvasása és beírása a memóriába, a ki/beviteli egység működésének vezérlése és a vezérlőegység időbeni működésének irányítása.

10. **A regiszterek osztályozása (hardver-szoftver, rendszer-felhasználói, általános-speciális) az INTEL (IA32) architektúra példáján.**

Hardver regiszterek: Pl.: I/O eszközvezérlők regiszterei
Szoftver regiszterek: A 8 általános célú regiszter, a 6 szegmens regiszter, a Flag regiszterek és az utasításszámláló együtt alkotják a végrehajtási környezetet, ami általános feladatok végrehajtására kell.
Rendszer regiszterek: Felhasználó által nem elérhető. Memóriavezérlés, megszakítás, végrehajtás, feladatkezelés, processzorkezelés a feladatuk Néhány elérhető programból.
Felhasználói regiszterek: PC, SP, FLAG, általános célú regiszterek
Általános célú regiszterek: felhasználási módjuk nem kötött, a gépi instrukciók argumentumaiban általánosan szerepelnek.
Speciális célú regiszterek: (korlátozottan használhatók) bizonyos instrukciók argumentumaiként nem szerepelhetnek (állapotregiszterek: CF, ZF, x87 FPU regiszterei, stb.).

11. **Az INTEL (IA32) architektúra általános regiszterei.**

Általános regiszterek
AX AKKUMULÁTOR (részeredmények tárolása. Leggyakrabban ezt használjuk, gyors)
BX BÁZISREGISZTER (indirekt címzésre használjuk, mutatót, pointert, címet tárolunk)
CX számláló regiszter (counter: ciklusokban az iterációk számát határozza meg, ciklusszervező))
DX adatregiszter (data r.: IO műveleteknél a portok címzésére használható, vagy olyan helyzetekben ha egy másik regiszter is kell egy feladathoz)
Vezérlő regiszterek
SP veremmutató (stack pointer)
IP Utasításszámláló (instruction pointer, a soron következő utasítás címének biztosítása CS-sel párban)
Szegmens regiszterek
CS kód szegmens (code segment : arra a szegmensre mutat, amelyben a következő utasítás található)

DS adat szegmens (data segment : az adatok tárolására szolgáló szegmens címét tartalmazza)
ES másodlagos adatszegmens (extra)
SS verem szegmens (stack)

12 Az IA32 flag regiszterek osztályozása, szerepe.

carry-flag (CF): átvitelbit, az utolsó művelet eredményének legfelső bitjét tárolja, ha túlcsordulás van.
overflow flag (OF): előjeles túlcsordulás
parity-flag (PF): paritás, páratlan paritás ellenőrzés
auxiliary flag (AF): segédátvitel, 3. és 4. bit közti átvitelt nézi
zero flag (ZF): akkor 1 az értéke, ha a vizsgált eredmény 0
sign flag (SF): előjel ellenőrzés
interrupt flag (IF): megszakítás engedélyezés
trap flag (TF): lépésenkénti futtatás

13 A CPU állapotai: rendszer (kernel, supervisor), illetve felhasználó (user, program) mód. Privilegizált utasítások.

A kernelmód arra szolgál, hogy futtassa az operációs rendszert és lehetővé tegye az utasítások végrehajtását. A felhasználói (user) mód feladata az alkalmazások futtatása, de nem engedi bizonyos érzékeny utasítások végrehajtását (mint például a közvetlen beavatkozást a cache-be).
Minden program a processzor user módjában fut. Ha a CPU user módban fut, bizonyos gépi utasításokat nem hajlandó végrehajtani -> (privilegizált utasítások) Azokat az utasításokat nevezzük privilegizált utasításoknak, amiket csak rendszer módban szabad végrehajtani. Privilegizált utasítások többek között a CLI, IN, OUT utasítások és a védett memóriacímek írása, olvasása. Ha egy alkalmazás privilegizált utasítást akar végrehajtani, akkor védelmi hiba lép fel, és meghívódik az operációs rendszer hibakezelő eljárása.

14 Az utasítás cím / program számláló regiszter funkciója. A gépi utasítások végrehajtásának mechanizmusa. A vezérlésátadás logikája.

A programszámláló regiszter (PC) tartalmazza azt a memóriacímet, ahonnan a soron következő utasítást kell venni. A programszámláló regiszter minden utasítás elővételekor automatikusan inkrementálódik, így mindig a soron következő utasításra mutat. Egy utasítás végrehajtása alapvetően az alábbi részekből tevődik össze:
Utasítás előkészítés, utasításlehívás: A processzor ebben a fázisban a következő utasítás memóriacímét, amelyet az utasításszámláló regiszter (PC) tartalmaz, átviszi a memória címregiszterébe
Utasításszámláló regiszter tartalmának növelése. A PC tartalmának automatikus növelésével előáll a következő utasítás tárolóbeli helyének memóriacíme.
Műveleti kód értelmezése, az operandus címének meghatározása. A processzor a műveleti jelrész dekódolásával meghatározza, hogy milyen utasításokat kell végrehajtania, valamint az utasítás címrésze alapján meghatározza a művelethez használandó operandus(ok) címét.
Adatok előkészítése a művelet elvégzéséhez. A központi egység az előzőekben kidolgozott cím alapján kikeresi az operandus(ok)at a memóriából és az utasítás által meghatározott helyre, amely az esetek többségében az aritmetikai egység akkumulátora (AC), de lehet más regiszter is.
Végrehajtás. Megtörténik az utasítás által kijelölt feladat elvégzése az előkészített operandussal.
Az eredmény elhelyezése. A központi egység a kapott eredményt elhelyezi az előírt helyre, amely többnyire az akkumulátor. Ezután újrakezdi az utasítás feldolgozást.
Vezérlésátadó utasítások
Feltétel nélküli és feltételes vezérlésátadó utasítások, azaz ugró utasítások.
Feltétel nélküli ugrásnál az utasításban szereplő címmel tölti fel a processzor az utasításszámláló regiszter tartalmát, amely a következő utasítás címe lesz és a program innen folytatódik. Feltételes ugró utasításnál a műveleti jelrész által előírt feltétel teljesülése esetén adódik át a vezérlés az operandusban megadott címre és a program az ott található utasítással folytatódik. Ha az operációs kód által előírt feltétel nem teljesül, a program a soron következő utasítással folytatódik.
Szubrutinhívó utasítások

Szubrutin (alprogram) hívó utasítások. Ezen utasítások feltétel nélküli vezérlésátadó utasítások, de megőrzik annak a helynek a címét, ahonnan a vezérlésátadás történt, így az alprogram végrehajtódása után a vezérlés visszatér az eltárolt címre és a program onnan folytatódik.

Ciklusszervező utasítások

Iteráció (hurok, vagy ciklus) utasítás. Az utasítás meghatározott programutasítások végrehajtását segíti elő az utasításban szereplő feltétel teljesüléséig.

15 A verem, mint speciálisan kezelt memória (rész).

A stack (LIFO, verem tár, zsák memória) a RAM memóriának egy speciálisan kezelt része, melynek a program végrehajtása során fontos szerepe van. A kezelésének lényege, hogy az adatokhoz a beírással ellentétes sorrendben lehet hozzáférni, vagyis mindig a legutoljára beírt adatot lehet legelőször kiolvasni. A beírás, ill. kiolvasás címét a stack pointer (SP) tartja nyilván.

Két olyan utasítás van, amely csak a stackbe való írást (PUSH forrás) ill. az onnan történő olvasást (POP cél) szolgálja.

A stack pointer értéke a legtöbb CPU-nál egy-egy adat betétele során csökken (a tartomány a csökkenő memóriacímek felé hízik), de van ellentétes példa is. A POP és PUSH utasítások által egyszerre betett, ill. kivett adat byte-ok száma CPU és utasítás függő.

16 A megszakítási rendszer, a CPU által végrehajtott lépések egy megszakítás elfogadása során, megszakítási vektor.

- 1: megszakítási kérelem fogadása
- 2: elbírálás
- 3: végrehajtás alatt álló program környezetének mentése
- 4: megszakítást kérő rutinnak megfelelő prioritás beállítása
- 5: megszakítás végrehajtása
- 6: az eredetileg futó folyamathoz tartozó prioritási szint visszaállítása
- 7: az eredetileg futó folyamat környezetének visszatöltése

Minden eszközhöz tartozik értelemszerűen egy-egy program, mely az eszközt kiszolgálja. Ezeket a programokat a gép indulásakor az operációs rendszer, vagy BIOS betölti a memóriába. A betöltött programok kezdőcímét nevezzük megszakítási vektornak, és azt a táblázatot, mely tartalmazza ezeket a címeket, megszakítási vektortáblának. Az eddig felvázolt megszakítást **hardveres megszakításnak** is nevezzük, mivel a folyamatot egy hardver elem váltja ki.

17. Megszakítások típusai (osztályai).

Szinkron - aszinkron megszakítások

Azon megszakítások, amelyek a programnak ugyanazon adatokkal való végrehajtása során mindig ugyanott lépnek fel, szinkron megszakításoknak nevezzük. Ilyen például az integer túlcsordulás.

Az aszinkron események viszont véletlenszerűen következnek be. Például az I/O egység kérte megszakítások, a hardver-hibák.

Az utasítások végrehajtása között fellépő megszakítások az éppen végrehajtott utasítás eredményeképpen következnek be (például túlcsordulás, page fault, tárvédelmi hiba). A kezelése rögtön az utasítás végrehajtása után elindulhat, s a programfutás eredményessége aztán a kezelés eredményétől függ.

Az utasítások végrehajtása közben fellépő megszakítások valamely utasítás végrehajtása alatt (tehát nem az utasítás-végrehajtási ciklussal szinkronban) merülnek fel. Ilyenek például a hardver-megszakítások. Ekkor az esetek többségében először befejezésre kerül az éppen végrehajtás alatt álló utasítás, s csak utána kezdődik meg a megszakítás kiszolgálása.

A felhasználó által explicit kért megszakítás például az operációs rendszer szolgáltatásának meghívása, a nyomkövetés vagy az utasítás töréspont. Véletlenszerűen fellépő

A felhasználó által nem kért megszakítás például az integer túlcsordulás, az I/O egység megszakítás, a hardver hiba.

A megszakított program folytatódik például I/O egység igénye alapján történő megszakítás, operációs rendszer szolgáltatásának meghívása esetén. A megszakított program futása befejeződik hardver hiba esetén.

Felhasználó által maszkolható vagy nem maszkolható megszakítások

A felhasználó által maszkolható például a nyomkövetés, a töréspont. Nem maszkolható viszont az I/O egység megszakítási kérése, az operációs rendszer szolgáltatásának meghívása.

Megszakítások típusai:

a. Megszakítás (interrupt),
b. Kivétel (Exception) például memória elfogyása, rendszererőforrás problémák, fájlhibák, 0-val osztás stb.
c. Nem maszkolható megszakítás (Non Maskable Interrupt) Olyan megszakítás, amelyet nem lehet letiltani, és amelynek kiszolgálását semmilyen körülmények között sem szakíthatja meg más kiszolgálási folyamat
d. Csapda (Trap)

Megszakítás esetén váltás történik a supervisor módba és a vezérlés egy megadott memóriaterületen lévő lekezelő rutinhoz kerül

18. <u>Az INT (SVC, TRcc) szoftveres megszakítás működése és szerepe az operációs rendszer szolgáltatások elérésében.</u>

Létezik az úgynevezett **szoftveres megszakítás** is, mely egy utasítás hatására váltja ki a folyamatot (pl. INT 134), csak értelemszerűen ehhez nem kell a megszakítás vezérlő. A szoftveres megszakítás folyamatát a fentebb leírt ***(16.tétel)*** 3.-7. lépések végrehajtása jelenti. Szoftveres megszakításokkal tipikusan a BIOS vagy a DOS szolgáltatásait érhetjük el. A szoftveres megszakításhoz a processzornak saját parancsa van az utasításkészletben.

19. <u>Eljárások, eljáráshívás. A verem szerepe, hívási konvenciók, paraméterátadás.</u>

Eljárás: Eljárást akkor készítünk, amikor egy részfeladatot többször végre kell hajtani a program különböző részein. Az eljárások használatának igazi előnye akkor jelentkezik, amikor nagyobb programokban egy-egy utasítássorozatot jó néhányszor meg kell hívni. Az eljárások használata áttekinthetőbbé teszik a programot, és az esetleges hibák javítása is egyetlen helyen elvégezhető, mégpedig az eljárás deklarált helyén.

Eljáráshívás: Ez az egyoperandusú utasítás szolgál az eljárások végrehajtására, más szóval az eljáráshívásra (procedure call). Működése: a verembe eltárolja az IP aktuális értékét (tehát szimbolikusan végrehajt egy PUSH IP-t), majd IP-t beállítja az operandus által mutatott címre, és onnan folytatja az utasítások végrehajtását. A célcímet a JMP-hez hasonlóan 16 bites előjeles relatív címként tárolja.

A verem olyan lineáris adatszerkezet, amelynek csak az utolsó elemével végezhetünk műveleteket. A verem (LIFO: Last In First Out), olyan tárolószerkezet, mely megőrzi a tárolás sorrendjét. Mindig csak az utoljára behelyezett elemet lehet kivenni. A verem szerepe, hogy félretegyünk bizonyos információkat későbbi felhasználás céljából, paramétereket, értékeket adjunk át. (Pl. operációs rendszer megszakítás-kezelés.) A verem deklarációjának tartalmaznia kell a verembe kerülő elemek típusát.

Hívási konvenciók

Call: eljáráshívó utasítás

Jump: eljárásra ugró utasítás

Mind a kettő annak az eljárásnak adja át a vezérlést, amit utána írunk. Pl.: Call ciklus

Paraméterátadások:

Kétféle paraméterátadási mód van: cím szerinti, érték szerinti.

Címszerinti paraméterátadás

Az eljárás hívásakor a paraméternek a címe kerül átadásra. A szubrutin a cím alapján tudja módosítani az átadott paraméter értékét úgy, hogy a megváltoztatott érték a hívó programba való visszatérés után is megmarad.

Értékszerinti paraméterátadás

Az eljárás hívásakor a szubrutin is helyet foglal a memóriában a változónak, ahova a paraméterként megkapott értéket bemásolja. Ha a függvény vagy eljárás megváltoztatja ennek a paraméternek az értékét, akkor a hívó programba való visszatérés után a megváltoztatott érték nem kerül vissza, hanem elveszik. Értékszerinti paraméterátadáskor a paraméter lehet konstans vagy kifejezés is.

20. A gépi utasítások operandusai, címzési módok.

Az utasítások operandusa az a hely (vagyis egy cím), ahol a programot folytatni kell, a feltételtől függően. Ha a feltétel nem teljesül, akkor a program a következő utasítást hajtja végre, tehát a végrehajtás folyamata nem módosul. Maga a címoperandus lehet literális – vagyis címke vagy címke kifejezéssel megadott cím – vagy regiszterben kijelölt, esetleg valamilyen indirekció. Továbbá a feltételes ugrásoknál előfordul implicit operandusú megoldás, vagyis amikor az ugrási címet nem adjuk meg konkrétan. Ilyen elágazó utasítás esetén a végrehajtás mindig a (program) sorban következő **vagy** a következő utáni utasításra helyeződik át – az elágazástól függően.

Címzési módok: (A címzési módokat hívják még operandus megadási módoknak is.)

Közvetlen címzés	**MOV AX, 42**	**konkrét érték (konstans)**
Direkt	MOV AX, [1234h]	; adott címen lévő érték (változó vagy konstans)
Regiszter	MOV AX, BX	változók közötti értékadás
Regiszter-indirekt	MOV AX, [BX]	pointer szerinti értékadás
Indexelt	MOV AX, [BX]4	Tömbindexelés
Bázis-index	MOV AX, [BP][SI]1234h	több dimenziós tömb
Verem	PUSH AX; POP AX	lokális változók használata

21. Az IA32 memóriacímzés alapelvei (szegmens, offset), szegmentálási módok.

Minden szegmens a processzor lineáris címtartományában helyezkedik el. Ahhoz, hogy egy adott szegmensben egy adott bájtot elérjünk logikai címet (ezt far-pointer-nek, távoli mutatónak is nevezik) kell használnunk. A logikai cím két részből a szegmens kiválasztóból és az ofszetből áll. A szegmens kiválasztó (pontosabban a leíró) egy egyedi azonosító a szegmens számára. A szegmens kiválasztó végül is egy index a leíró táblázat egy elemére a kívánt szegmens leíróra. Így a szegmensleíró meghatározza a szegmens méretét, a hozzáférési jogokat, a szegmens privilégium szintjét, a típusát és az első kezdő bájtot a lineáris címtartományon (ezt a szegmens kezdőcímének nevezzük). Az ofszet határozza meg a szegmensen belül az elérni kívánt bájt helyét. (Az ofszet simán hozzáadódik az előbb elmondott szegmens kezdőcímhez, lesz lineáris cím a processzor lineáris címtartományában.)

Módok:

8086 (régi szegmentálás)
– Max 1 MB memória
– 64 KB szegmensek
– 16 x 64 KB = 1MB
– Bármely cím elérhető, csak be kell tölteni a szegmens regiszterbe
• CS : kód szegmens szelektor
• DS : adat szegmens szelektor

80386, védett módú címzés
– 4 GB memória
– A memória nem elérhető szabadon
– A szegmens bármekkora lehet
• 1 byte
• 64 KB
• 4 GB

22. IA32 memória operandusok címzése (bázis, index, skála, displacement).

A forrás és cél operandusok a memóriában a szegmens szelektorra és az ofszetre hivatkoznak. A szegmens szelektor leírja a szegmens tartalmát az operandus és az ofszet (byte-ok száma a szegmens elejétől az operandus első bájtjáig) leírja a lineáris vagy az effektív címét az operandusnak. Az ábra a memória operandus címét mutatja:

Segment selector		Offset (vagy lineáris cím)	
15	0	31	0

Az offset része egy memória címnek, mely meg lehet adva közvetlenül, mint állandó érték (ezt hívják displacement-nek, elhelyezkedésnek), vagy egy címszámításon át eljuthatunk egy vagy több komponenshez az alábbiak közül:

- Displacement – egy 8-16-32 bites érték
- Base Az értéke egy általános regiszterben van
- Index Az értéke egy általános regiszterben van
- Scale factor – az értéke 2,4,vagy 8 ami szorozva van az index értékével.

Offset vagy Effektív cím értékének kiszámítása: Bázis + (Index*Skála)+Elhelyezkedés

Megj: ESP-t nem lehet indexként használni.

23. <u>Az IA32 gépi utasítások szerkezete, prefixumok</u>

Utasítások: számítógép számára érthető tovább nem bontható elemi lépések

Utasítások tárolása

Adatokkal azonos módon 2-es számrendszerben

Utasítások általános felépítése:

MK - Műveleti kód (Operation Code)

CM - címzési mód cím kiszámítás módjára utaló rész

címrész operandusok helyét kijelölő rész

Külön utasítások létezhetnek a byte, szó (2 byte) duplaszó (2 szó) stb. tartományok elérésére, de lehetséges, hogy ezt előtagok – más néven prefixum – segítségével kell elérni. Ezek – formailag – az utasítás előtt helyezkednek el, és egy utasítás előtt egyszerre több is lehet belőlük (a maximális szám processzorfüggő). Tehát a memóriareferens utasítások kimenetelét illetve lezajlását különböző előtéttagok befolyásolhatják:

- ismétlő prefix – ismétlés adott feltétel fennállásáig
- operandusméret prefix – az elérni kívánt memóriarekesz nagyságát befolyásolja
- címmódosító prefix – az elérni kívánt memóriarekesz címének értelmezését befolyásolja
- szinkronizációs prefix – a CPU és az FPU közötti szinkronizáció céljára (a 386-os PC-nél már hardveresen működik eme mechanizmus)
- buszlezárás prefix – az adatmódosítás biztonságát (konzisztencia) növeli, például többprocesszoros rendszerekben

24. <u>Az IA32 gépi utasítások osztályozása (adatmozgatás, aritmetika, logika, vezérlésátadás, egyéb,…)</u>

A ZH segédlet

25 <u>A makró-assembler alapvető szolgáltatásai, szimbolika, direktívák.</u>

Masm: a Microsoft által írt makró assembler Intel processzorokhoz, DOS/Windows platformra. Használatával szöveges assembly forrásfájlokból készíthetünk bináris futtatható exe/com fájlokat.

Direktívák:

- A programstruktúra definiálására: SEGMENT, ENDS, END, GROUP, ASSUME, ORG, EVEN
- utasításkészlet meghatározására: P8086, P186, P286, P386, .8086, .186, .286, .386
- Szimbólum definiálására: DB, DW, DD, DQ, DT, LABEL, EQU, =
- Szubrutinok: PROC, END
- Külső szimbólumok definiálása: EXTRN, PUBLIC
- Makrók: MACRO, ENDM, LOCAL, EXITM, PURGE
- Kódismétlő makrók: REPT, IRP, IRPC

- Feltételes fordítás: IF, IFE, IF1, IF2, IF(N)DEF, IF(N)B, IFIDN, IFDIF, ELSE, ENDIF
- Adatszerkezetek: STRUC, ENDS, RECORD
- Fordításvezérlés: INCLUDE, .RADIX, %OUT, NAME, TITLE, SUBTTL, PAGE
- Listázás: .XLIST, .LIST

Szimbólumok: a konstans azonosítók, változók, címkék, szegmensek, eljárások nevei. Szimbólum azonosítójának mindig betűvel kell kezdődnie, azután tartalmazhat számjegyet, betűket (csak az angol ábécé betűit), aláhúzás jelet (_), dollárjelet ($) és "kukacot" (@).Speciális szimbólum az egymagában álló dollárjel, ennek neve pozíció-számláló (location counter). Értéke az aktuális sor szegmensbeli (avagy szegmenscsoportbeli) ofszetje.

26 Szuperskalár processzorok, cső (pipe).

A futószalag-technika napjainkban az utasítás-feldolgozás szabványos módja. A jelenlegi processzorok gyakorlatilag mind alkalmazzák. A futószalag technika a legáltalánosabb módja az utasítás-feldolgozó egység gyorsításának. A nem RISC, nagy teljesítményű gépekben is alkalmazzák ezt a technikát, de a RISC gépek legtöbbjére ez jellemző. A futószalag fokozatai számának növelésével azonban a teljesítmény nem sokszorozható meg a végtelenségig, két ok miatt: a fokozatok a gyakorlatban különböző időt igényelnek, így komoly időveszteségek léphetnek fel; a függőségek miatt. Az egyik legelső, és legerősebb technika a teljesítmény javításában az utasítás pipeline-ok alkalmazása. A pipeline-ok növelik a teljesítményt azáltal, hogy lehetővé teszik több utasítás párhuzamos végrehajtását a processzorban. A processzor elkezdené egy új utasítás dekódolását (1. lépés), míg az utolsó még az eredményre vár. Ez lehetővé teszi, hogy 4 utasítás legyen „repülés közben" ugyanabban az időben, ezáltal a processzor 4× olyan gyorsnak tűnik. Bár egy utasítás végrehajtása ugyanannyi időt vesz igénybe (még mindig 4 lépésből áll) a CPU egészét tekintve sokkal gyorsabban „fekteti el" az utasításokat és sokkal magasabb órajelen futtatható.
A szuperskalár kifejezés arra mutat rá, hogy a lineáris technikán túlmutató módon, az utasítás-feldolgozás újraszervezése révén érünk el igen jelentős teljesítmény-növekedést. A szuperskalár processzorok egynél több végrehajtó egységgel rendelkeznek. Az Intel családon belül például az első szuperskalár processzor a sima Pentium volt: 2 db végrehajtóegységgel rendelkezett (az U és a V pipe). Több utasítás végrehajtása 1 ciklus alatt.

27. CISC és RISC proccesszorok, SIMD megoldások.

Risc=Reduced Instruction Set Computer (Csökkentett utasításkészletű számítógép)
Cisc=Complete Instruction Set Computer (Teljes utasításkészletű számítógép)
A Risc utasításkészlete azért kisebb, mert az utasításkészlet csak a leggyakrabban használt, egyszerűbb gépi utasításokat tartalmazza. Míg a Cisc több 10 utasítást tartalmaz, addig a Risc sokkal kevesebbet. A RISC filozófia bevezetése rengeteg vitát eredményezett az érdemeiről. A kulcskérdések egyike a teljesítmény, mivel a RISC processzorokat úgy tartják, hogy túlszárnyalják a CISC-eket. S valóban, a mérések alapján Assembly nyelvnél 30-50%-kal, magas szintű programnyelvnél, mint pl. a C, 2-3szor gyorsabb. A következő szám a relatív kódméret. Mint várható, a RISC kód terjedelmesebb, mert az utasítások egyszerűbbek, azaz azonos funkcióhoz több gépi kódú utasításra van szükség: 40-50% statikus kódnövekedés: ennyivel több a gépi kódú utasítások száma; 10-30% dinamikus kódnövekedés: ennyivel nő a ténylegesen felhasznált bájtok száma. A RISC filozófia fő hozzájárulásának a RISC I processzor nagy regiszter-készletét tartják, különösen pedig a többszörös, átfedő regiszterkészletet. Az a RISC javára szolgál, hogy a vezérlési rész csökkentése lehetővé tette, hogy a lapkán olyan nagy regiszter állomány jöhessen létre, amilyen addig nem volt lehetséges.
A SIMD típusú számítógép egyetlen utasításfolyammal többszörös adatfolyamot dolgoz fel. Ezek a számítógépek több párhuzamos, egyidejű működésre képes műveletvégző egységet (ALU)

tartalmaznak. Vektorműveleteket képesek végrehajtani, gépi utasítás szinten (például a Pentium III processzor 3D grafikus utasításai).
SIMD megoldások: A szemcsézettség
Meghatározza a viszonyt a feldolgozóelemek száma és az adatkészlet párhuzamossága között.
Finom szemcsézettségű rendszerek: Minden feldolgozóelemhez csak kevés számú adatelem tartozik
Durva szemcsézettségű rendszerek: Minden feldolgozóelemhez sok adatelem tartozik
Az ideális az 1:1-es megfeleltetés lenne, de még nem megvalósítható. Másik probléma a különböző alkalmazásokban használt adatkészletek eltérő mérete. Nehéz rugalmasan bővíthető SIMD rendszereket készíteni. Az adatelemenként elvégzendő számítások bonyolultsága hatással van a feldolgozóelemek típusára és bonyolultságára. Ez pedig technológiai és gazdaságossági szempontból hatással van a maximálisan elérhető párhuzamosságra. A megoldások általában a két végletet célozzák meg: Finom szemcsézettség egyszerű feldolgozóelemekkel, és durva szemcsézettség bonyolult feldolgozóelemekkel

28. „Többmagos" processzorok, szálak (multithreading).

Az órajelek folyamatos növelésével a processzorfejlesztések elértek egy olyan határt, hogy új megoldást kell találni a további fejlődéshez, mivel 4 gigahertz felé a processzorok hatalmas hőt termelnek. Ennek eredményeként születtek meg az első kétmagos processzorok. Tehát az elv csak annyi mindössze, hogy a gyorsulás mellet ne termeljenek annyi hőt. Ezeknek is jelentek meg újabb megoldásai, pl. a hyper threading amely dupla (vagy akár több) annyi processzorszálat vetít az operációs rendszernek. Tehát egy 4 magos Intel processzor mely hyper threading technológiával rendelkezik, azt az operációs rendszer 8 (vagy több) magosnak látja. Ezzel a technológiával CPU több feladatot tud egyszerre végrehajtani hőt és energiát megtakarítva.