**Architektúrák - licensz vizsga 2021**

**1. A Neumann architektúrájú processzorok minimális regiszter készletének funkcionális leírása.**

* ***Vezérlőegység (CPU):*** *A gép irányításáért, a folyamatok vezérléséért felelős. A számítógépek nagyon bonyolult feladatokat is végre tudnak hajtani, ha azok olyan egyszerű lépések sorozatára vannak bontva, melyeket a számítógép értelmezni tud. A központi feldolgozó egység a számítógép agya. A központi egység fő feladata végrehajtani a memóriában tárolt utasítás-sorozatot.*
* ***Aritmetikai és logikai egység (ALU):*** *Az adatok feldolgozásáért felelős, egy vagy két operandusú matemetikai és logikai műveleteket végez.*
* **Regiszterek**: Átmeneti tárolóhelyek a processzor belsejében. Gyorsabban elérhetőek mint a memória.
  + **AC** (akkumulátor regiszter): általános célú regiszter, műveletekhez szükséges operandusok egyikét, valamit azok eredményét tárolja
  + **AR** (adat regiszter): adatátvitelt biztosít a memória és a CPU belső alkotóelemei közt (olvasásnál ide kerül be először a memóriából, írásnál ide kerül bele, majd innen íródik ki)
  + **PC** (program számláló): a következő utasítás címét tárolja
  + **CR** (cím regiszter): a memória fele közvetített (utasítás vagy operandus) címét tárolja (ide íródik be a cím, amelyen az következő utasítás, vagy éppenséggel az adat található, ezzel kerül megcímzésre a tároló)
  + **UR** (utasítás regiszter): dekódolja az utasítást, felosztja utasítás kódra (ez megy a PU fele) illetve operandus címre (ha van, pl. melyik címen van a szükséges adat)
  + **SP (stack pointer)** – verembe mutatja a kovetkezo szabad terulet cimet
  + **FR (flag register)** – allapotjelzo bitteket tartalmaz, zero, carry, interrupt enable
* ***PU*** *(processing unit): többállapotú automata, az egész CPU-t összehangolja, az általa létrehozott kontrolljelek vezérlik az összes részegység működését*

**2. A Neumann architektúrájú processzorok vezérlő egységének funkcionális leírása.**

**-** A vezerlo egyseg iranyitja az ALU-t, memoriat, I/O eszkozoket

**-** A Vezérlő egység irányítja, tehát az utasítások végrehajtását, annak közös (kiolvasás és dekódolás – Fetch and Decode) illetve specifikus (műveletvégzés) fázisait.

**-** a központi egység fő feladata végrehajtani a memóriában tárolt utasításokat

- egy utasítás-végrehajtás ciklus két fázisból tevődik össze:

- az utasítás előhívása a memóriából és ennek egy belső regiszterbe való betöltése (fetch and decode)

- az utasítás végrehajtása (execute)

- az utasítás végrehajtása tartalmazza az utasítás dekódolását, az operandusok betöltséét, a múvelet végrehajtását, és ha szükséges, az eredmény visszaírását a memóriába

- a vezérlő egységnek kell biztosítani az utasításokhoz szükséges vezérlőjelek sorozatát

- a vezérlőjelek engedélyezik, vagy lezárják az adat útvonalát, viszont ez több útvonalon is átvezethető

- azt, hogy melyik útvonalon halad az adat, az utasítás határozza meg

**-** a vezerloegyseg figyeli a megszakitas kereseket

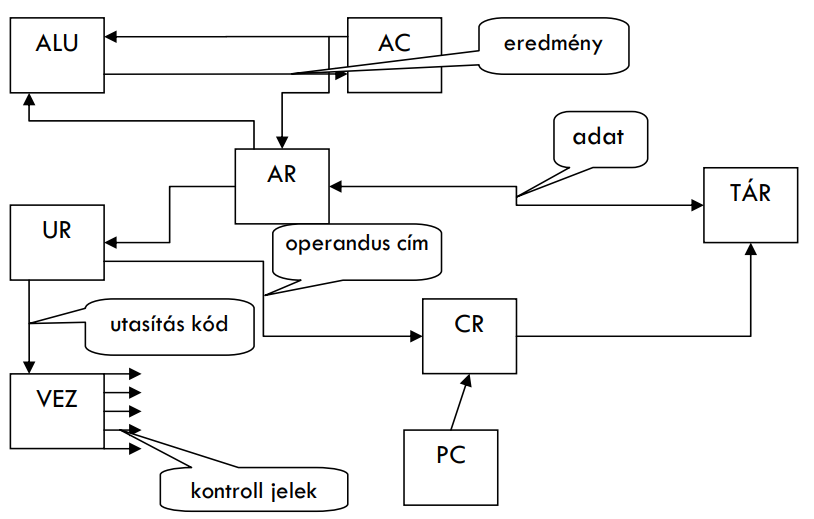
A vezérlőegység két funkciója azonosítható:

* **Az utasítások szekvencia előállítása** (z utasítások sorba állítása) (azon metódusok, amelyek kijelölik a következő végrehajtandó utasítást)
* **Az utasítások értelmezése** (metódusok a dekódolt utasítás vezérlőjeleinek előállítására)

**Az utasítások szekvencia előállítása:**

* Az esetek nagy többségében egy program valamely utasítását az eggyel nagyobb memóriacímen levő utasítás követi.
* Az előbbi érvből adódóan, a legegyszerűbb megoldás az utasítássorozatok előállítására egy programszámlálónak az alkalmazása, amely az aktuális végrehajtandó utasítás dekódolása után automatikusan inkrementálódik eggyel. Ebben az esetben nem szükséges az utasításkódban meghatározni (megadni) a következő utasítást, csak speciális esetekben, mint: ugróutasítás, függvényhívás, megszakítás.
* Az ugrási utasítás esetében a vezérlőegység biztosítja a programszámláló párhuzamos feltöltését az utasításkódban meghatározott címmel.

**3. A processzorok belső sínjeinek – az adatút – kialakításának lehetőségei.**



A sín különböző eszközök közötti elektronikus pálya.

A központi feldolgozó egység a rendszerhez (a rendszer többi eleméhez) a három sínrendszeren keresztül kapcsolódik:

* **Adat-sín**
  + egy kétirányú sínrendszer, és az adatok és utasítások átvitelére szolgál a processzorból és a processzorba
  + általában a processzor által meghatározott számú vezetéket tartalmaz
  + rendszerint az adatsín 8, 16, 32 vagy 64 bit szélességű
* **Cím-sín**
  + a processzor ezen a sínrendszeren keresztül küldi ki a memória vagy periféria címét, ahonnan az adatokat beolvassa /ahova az adatokat kiírja
  + a címsín szélességét minden esetben a processzor határozza meg, amiből következtetni lehet az elérhető fizikai memóriaterület méretére is
  + általában a memória és az I/O eszközök használhatják ugyanazt a címtartományt, de természetesen meg kell egymástól különböztetni ezeket
* **Vezérlő-sín** – A vezérlősín jelei lehetnek:
  + adatmozgatásra használt vezérlőjelek- vezérli a memória és a ki/bemeneti perifériák elérési műveleteit
  + megszakításkérő vonalak – a periférikus ki/bemeneti egységektől a processzor felé
  + a megszakításkérés elfogadása- a processzortól a ki/bemeneti egységek felé
  + a sínrendszer arbitrálása (elbírálása) – ha egy rendszeren belül több processzor található, ezen vezérlőjelek segítségével bírálják el a processzorok egymás között, hogy ki fogja vezérelni a sínrendszert.

A síneket funkcióik szerint osztályozhatjuk. Használhatjuk őket CPU-n belül, hogy adatokat vigyünk be az ALU-ba vagy onnan ki, illetve a CPU-n kívül, hogy összekössük a CPU-t a memóriával.

**4. Függvényhívási utasítás végrehajtásához szükséges CPU alegységek és azok feladatai.**

- függvényhívás esetén, a függvényhívást követő utasítás címét (**PC+1**) a vezérlőegység automatikusan elmenti a verembe (**SP**), majd a programszámlálóba (PC) betöltődik a függvény kezdőcíme

**-** a visszatérés esetén, a vezérlőegység kiolvassa a verem tetejéről az elmentett utasítás címét, majd betölti a programszámlálóba

**-** ezen kívül, ha szükséges, olyan regiszterek (pl. **FR**) állapotai is elmentődnek, melyek állapotai szükségesek a korábbi állapot visszaállításához; ezek értékei a függvényből való visszatérés esetén ismét betöltődnek

**5. Az utasítás-formátum hatása a processzorok utasítás-készlet architektúra (ISA – Instruction Set Architecture) szintjének tervezésére.**

- a Neumann architektúrához tartozó utasítások másként vannak felépítve:

| Utasítás kód **|** Operandus cím **|**

**n m**

- a memória lokáció mérete **n+m**, a processzor pedig utasítást tud kezelni, memória lokációt lehet megcímezni

**6. Az operatív tárak szervezése. Lapszervezésű virtuális tár.**

* A számítógépek memóriája, felhasználás szerint 3 típusú lehet:
  + Belső memória (regiszterek): kicsi, gyors, ideiglenes tárolásra szolgál
  + Alap/központi memória: működés közbeni tárolásra szolgál, közvetlen kapcsolaton keresztül fér hozzá a CPU (sín, vezérlő jelek)
  + Háttér memória: nagy, lassú, hosszú távú tárolásra alkalmazzák
* Elérési mód szerint:
  + RAM: minden lokáció elérési ideje ugyanaz
  + Soros elérésű memória: az elérési idő változik a cím függvényében
* Fizikai felépítés alapján:
  + RAM
  + ROM
  + EPROM – optikai úton törölhető, újrapprogramozható
  + EEPROM – elektromosan törölhető, újraprogramozható
  + FLASH-ROM – gyors, nagyszámú törlés/újraírást támogató ROM memória
* a virtuális memória lényegében a címzéstartományt jelenti, ez nem más mind egy másodlagos memória (háttértár)
* a virtuális memórria egyenlő lapokra oszlik
* létezik egy laptáblázat, mely információt tartalmaz minden oldalról: az odlal címét a memóriában, be van-e töltve, és állapotbiteket
* egy virtuális cím egy lapcímből és egy eltolásból áll, a processzor a Memory Management Unit-on keresztül kikeresi a lapszámnak megfelelő információt, ha az alapmemóriában megtalálható, kiolvassa a kezdőcímét és hozzáadva az eltolást, kialakítja a fizikai címet
* ha nincs bent a lap, beolvasódik a háttértárról
* ha nincs elég hely a lap betöltésére, a legrégebben használt lap, kicserélődik a kérttel
* oda-vissza másolgassa - SWAP művelet

**7. Vektorizált megszakítás rendszert alkalmazó I/O adatátvitelek végrehajtásának lépései.**

* Megszakias vektor – minden periferianak van egy sajat megszakitas vektora, amely tartalmazza a megszakitas informacioit. Ilyenek pl hogy milyen periferiarol erkezett a megszakitas, milyen esemeny keletkezett, pl billentyuzetrol valo beolvasas eseten melyik gombot nyomtunk le
* CPU kiolvassa a megszakítás vektor táblázatból a lekezelő rutin címét, lementi a PC+1-et, illetve a FR-t
* Lefut a lekezelo rutin
* Visszater az azelotti feladathoz
* 2 utasitás végrehajtása között teszteli az iINT vonalat, ha aktiv akkor kéri a forrást,

**8. Közvetlen memória-hozzáférést (DMA) alkalmazó I/O adatátvitelek végrehajtásának lépései.**

* célja a háttértárolón lévő adat memóriába helyezése, minimális CPU időt felhasználva
* ez néhány egyszerű lépséből valósítható meg:
  + a CPU felprogramozza a DMA kontrollerét C darab byte átvitelére X címre
  + a DMA egy kérést intéz a háttértár kontrolleréhez, az adat átmásolására a memóriába
  + a kontroller inicializálja a DMA átvitelt
  + a kontroller elküld minden byte-ot a DMA kontrollerének, a DMA növeli a cél címét(++X), csökkenti a byte-ok számát(--C), ismétli míg C!=0
  + ha nincs több átvitelre váró adat, a DMA egy megszakítást küld a CPU felé, jelezvén, hogy az átvitel megtörtént

**9. Az utasítások végrehajtásának párhuzamosítása csővezeték struktúrák (pipe-line) alkalmazásával.**

* a lapozás és szegmentálás megnöveli az utasítások végrehajtásának idejét, mivel több mikorlépésre van szükség, valamit több memória hozzáférési ciklusra
* alapesetben is megkülönböztetünk 3 lépést: **fetch => decode => execute**
* ha gyorsítani akarjuk ezt a folyamatot, akkor tervezhetünk külön egységeket melyek 1-1 fázis tudnak elvégezni. Ezeket pedig pipeline struktúrába helyezzük, ekkor így alakul:

**fetch => decode => exec**

**fetch => decode => exec**

**fetch => decode => exec**

* ilyen módon elérhetünk egy gyenge párhuzmaosítást, illetve növeljük a sebességet is