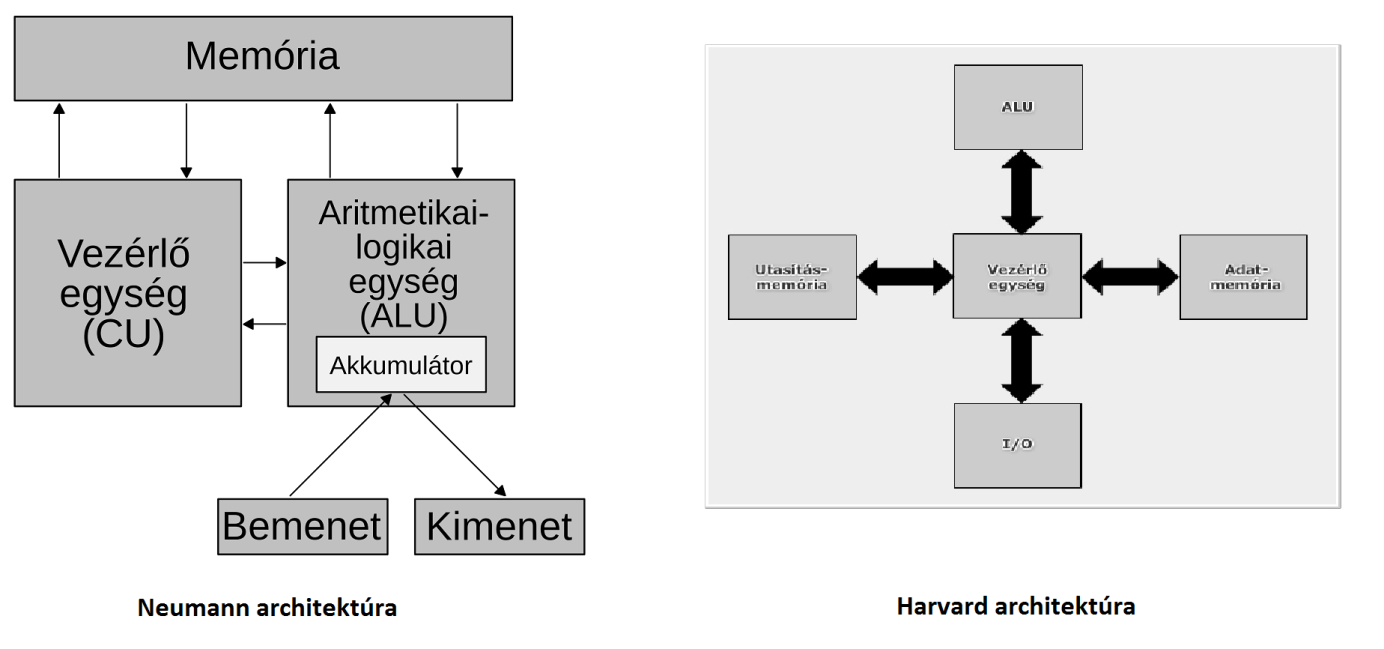
**ARM architektúra**

Az ARM architektúra nyílt szabvány, forráskódja szabadon módosítható ezért több gyártó is ARM processzorokkal szereli a készülékeit.

Az ARM architektúra Harvard architektúrára épül, amely valamelyest bonyolultabb, mint a Neumann architektúra. A következő ábrán megfigyelhető a két architektúra közti legfőbb különbség, ami nem más, mint az adat és programmemória elhelyezése.



Az ábrán is jól látható, hogy a Harvard architektúra esetében az utasítás és adatmemória külön helyezkedik el.

A utasításkészlet nélküli számítástechnika (NISC - No Instruction Set Computing):

* Magyarul utasításkészlet nélküli számítástechnika
* statikusan időzített: művelet időzítő és veszélyesség kezelő kezelve van a fordító által
* vízszintesen nanokódolt: a NISC nem rendelkezik semmilyen előredefiniált utasításkészlettel vagy mikrokóddal
* a fordító nanokódot generál, amely közvetlenül vezérli egy adott adatút funkcionális egységeit, regisztereit és multiplexereit

Előnyei:

* Egyszerűbb vezérlő: nincs hardverütemező, nincs utasítás dekódoló
* Jobb teljesítmény: még rugalmasabb architektúra, jobb erőforrás kihasználás
* Könnyebb tervezni: nincs szükség utasítás-készletek tervezésére

Az összetett utasításkészletű számítástechnika (CISC - Complex Instruction Set Computing)

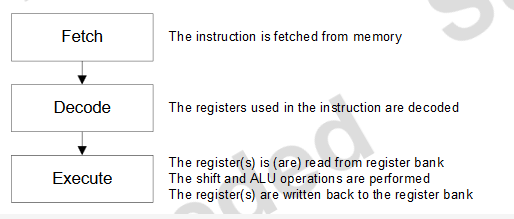
* bonyolult utasításokat tartalmaz
* utasításai általában összetettek, több elemi műveletet végeznek egyszerre
* a gépi programjai rövidebbek, jobban átláthatóak
* közvetlen memóriaelérés jellemző rá
* mikroprogramozott vezérlési eljárás: végrehajtás alatt lévő utasítás egy mikroprogramtárban lévő programot indít el
* rövidebb programok ellenére is viszonylag lassú futás jellemzi a bonyolult utasítások miatt

A csökkentett utasításkészletű számítástechnika (RISC - Reduced Instruction Set Computing)

* Betöltés/tárolás (load/store) architektúra: memóriaelérés csak *load* és *store* műveletek segítségével:
  + nincs olyan utasítás, ami a memóriaelérést (load/store) az aritmetikával kombinálja
  + minden műveletvégző utasítás regisztereket használ
* kisszámú, egyszerűsített címzési mód, egyáltalán nem használ indirekt címzést
* minden utasítás ugyanolyan hosszúságú
* átlapolásos utasítás végrehajtás (pipeline): egyértelműen különválasztja az utasítás eljárások egyes fázisait így több utasítást hajt végre a processzor párhuzamosan.
* az utasításokat lehetőleg 1 órajel ciklus alatt hajtja végre
* nagyszámú általános célú regiszter

Az ARM architektúra a következő [RISC](http://hu.wikipedia.org/wiki/RISC) képességekkel rendelkezik:

* Load/store architektúra
* Csak illesztett (aligned) memória elérés az ARMv6-ig
* Ortogonális utasítás-készlet
* Nagy, 16×32 bit regiszter-készlet
* Egységes, 32 bites utasítások (egyszerűbb dekódolás és pipelining, csökkentett kódsűrűség)
* Az alábbi ábrán egy ARM7TDMSI processzor utasítás végrehajtási pipelineja látható:



Utasítás beolvasása, fordítás majd végrehajtás.

* Többnyire egy-ciklusos végrehajtás
* Betöltés/tárolás (Load/store) architektúra
  + Azért lett kifejlesztve, hogy egyszerűsítsék a CPU felépítését, és növeljék a teljesítményét.
    - Memory wall: a CPU-k gyorsabbak, mint a memória
    - A memória elérés lassítja a CPU-t, ezért módosították az utasításkészletet, hogy több utasítást is memóriától függetlenné tegyenek
  + Az adatfeldolgozó utasítások csak a regiszterekhez tudnak hozzáférni
    - A regiszterbe tölti az adatot
    - Feldolgozza
    - A memóriába tárolja vissza
  + Sokkal hatásosabb, ha több regiszter elérhető
* Regiszter/Memória architektúra
  + Az adatfeldolgozó utasítások hozzáférhetnek a memóriához vagy a regiszterekhez
  + Regiszterek:
    - R0-12 általános célú regiszterek adatfeldolgozáshoz
      * R0-7: sok 16 bites utasításnak csak ezekhez van hozzáférése
      * R8-12: 32 bites utasításokkal lehet használni
    - Veremmutató (Stack Pointer - SP) (R13)
      * Lehet: Main Stack Pointer (MSP) vagy Process Stack Pointer(PSP)
      * Alapértelmezettként és Kezelő (handler) üzemmódban MSP-t használ
      * Fonal (thread) módban vezérlő (CONTROL) regiszter használatával lehet választani MSP és PSP között
    - Link Register-LR (R14)
      * tárolja a visszatérési címet, ha B&L utasítás lett meghívva
    - Programszámláló (Program Counter-PC) (R15)
  + Speciális Regiszterek
    - Programállapot regiszter (Program Status Register xPSR)
      * APSR – Alkalmazás regiszter (Application Register)
      * EPSR – Végrehajtó regiszter (Execution Register)
      * IPSR – Megszakítás regiszter (Interrupt Register) (csak olvasható)
      * Ezt a hármat egy kombinált regiszterként is kezelhetjük
    - Megszakítás/kivétel regiszter (Interrupt/exception register)
      * PRIMASK
      * FAULTMASK
      * BASEPRI
    - CONTROL
    - Floating point status and control register FPSCR
* Bájtsorrend (Endian)
  + A bájtsorrend a tárolási/hálózaton történő továbbítási sorrendet jellemzi. ARM esetében lehetőség van a bájtsorrendek közötti átváltásra, átkapcsolásra. Manapság a legelterjedtebb a little endian használata, de az ARM támogatja a little és a big endiant is:
    - Little endian: A növekvő bájtsorrendet jelenti, a „legkisebb” lesz legelöl (pl.: Intel x86)
    - Big endian: Az előző ellentetje lesz, azaz a „legnagyobb” lesz a legelső helyiértéken (pl.:Motorola processzorok)
* Tömörített utasításkészlet (Thumb)
  + Legtöbb utasítás 16 bit hosszúságú
  + Leginkább R0-R7 regiszterekhez van hozzáférése, de némelyiknek van a R8-R15 regiszterekhez is
  + Az ARMv7E-M csak tömörített utasításokat használ
  + Amikor a memória port, vagy busz kevesebb, mint 32 bitre van korlátozva, a tömörített utasítások jobb teljesítményt nyújtanak
  + Kevesebb funkció érhető el vele
  + A feltétel végrehajtás támogatva van 16 és 32 bitre is