Pannon Egyetem
Villamosmérnöki és Információs Rendszerek Tanszék

Digitális Rendszerek és Számítógép Architektúrák (BSc államvizsga tétel)

1. tétel: Neumann és Harvard számítógép architektúrák összehasonlító elemzése - számítógép generációk

Összeállította: Dr. Vörösházi Zsolt
voroshazi@vision.vein.hu

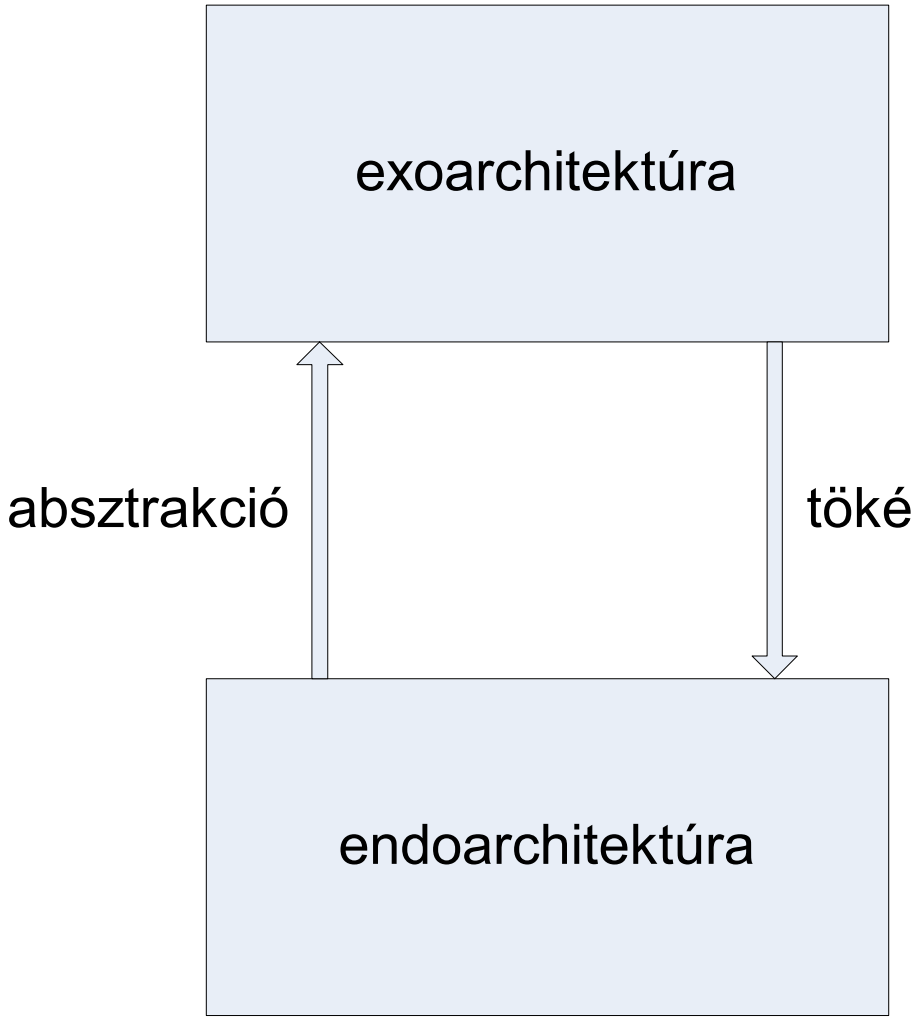
Kapcsolódó jegyzet, segédanyag:

- Tétel: Informatika / Programtervező Informatikus / Gazdaságinformatikus BSc alapszakos hallgatóknak (2012. május)
- Könyvfejezetek:
  - http://www.virt.uni-pannon.hu
    → Oktatás → Tantárgyak → Digitális Rendszerek és Sz.gép Arc. / Számítógép Architektúrák
  - Bevezetés: Számítógép Generációk (chapter01.pdf)

Alapfogalmak:

- **A számítógép architektúra** a hardver egy általános *absztrakciója*: a hardver struktúráját és viselkedését jelenti más rendszerek egyedi, sajátos tulajdonságaitól eltekintve
- **Architektúrális tulajdonságok** nemcsak a funkcionális elemek leírását, hanem azok belső felépítését, struktúráját is magába foglalják

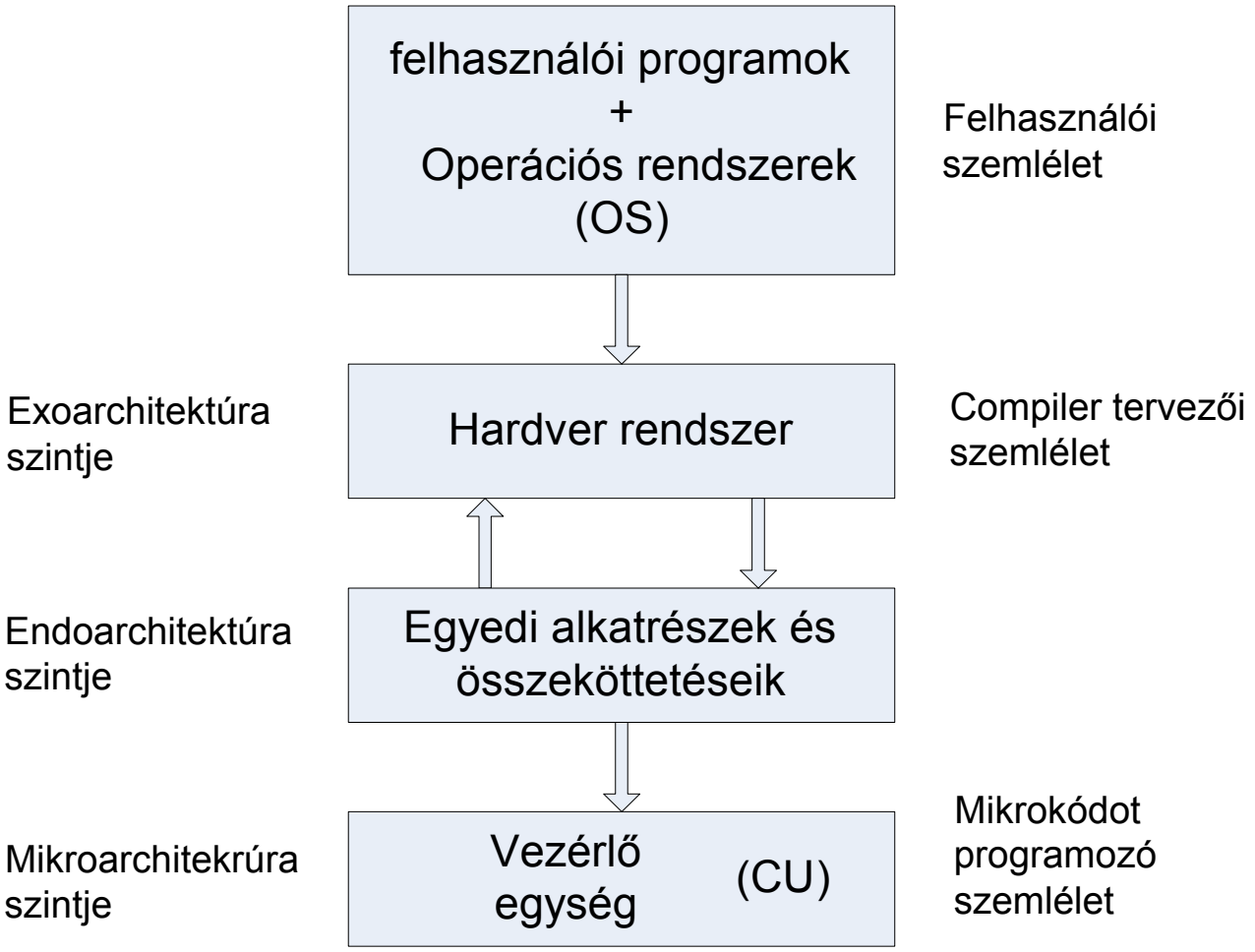
Exoarchitektúra – endoarchitektúra:



Compiler tervezői aspektus:
- utasítás készlet
- utasítás regiszterek
- címzési módok

Vezérlő egység tervezői aspektusai:
- jelek
- komponensek

Számítógép architektúra definíciója:



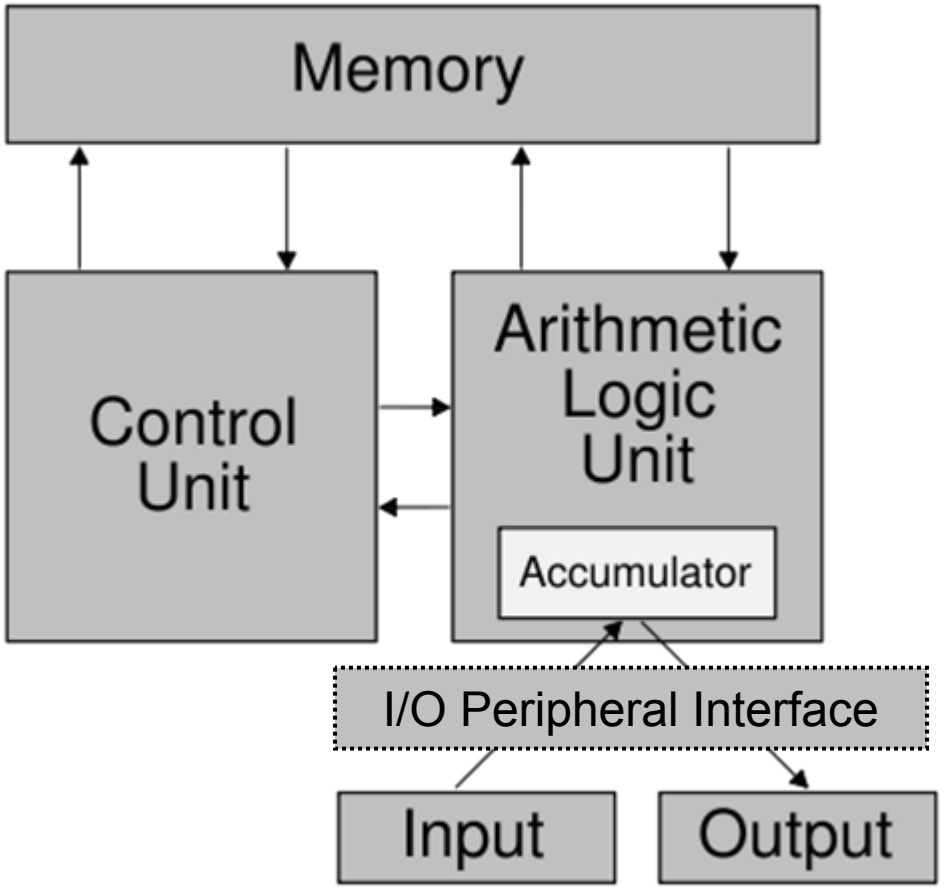
Számítógépes rendszerekkel szembeni tervezői követelmények:

- Aritmetika megtervezése, algoritmusok, módszerek elemzése, hogy a kívánt eredményt elfogadható időn belül biztosítani tudja
- Utasításkészlet – vezérlés
- A részegységek közötti kapcsolatok / összeköttetések a valós rendszert szemléltetik
- CFG, DFG a főbb komponensek között
- Számítógép és perifériák közötti I/O kommunikációs technikák

Neumann, Harvard számítógép architektúrák

A.) Neumann architektúra

- **Számítógépes rendszer modell:**
  - CPU, CU, ALU
  - Egyetlen különálló tároló elem (utasítások és adatok részére)
  - Univerzális Turing gépet implementál
  - „Szekvenciális” architektúra (SISD): single instruction→ single data működés



Von Neumann architektúra

- De Facto szabvány: „single-memory architecture”. Az *adat-* és *utasítás* címek a memória (tároló) ugyanazon címtartományára vannak leképezve (mapping). Ilyen típusú pl:
  - EDVAC (Neumann), egyenletmegoldó tárolt-programú gép
  - Eckert, Mauchly: ENIAC, UNIVAC (University of Pennsylvania) – numerikus integrátor, kalkulátor
  - A mai rendszerek modern mini-, mikro, és mainframe számítógépei is ezt az architektúrát követik (operatív tárak/main store szintjén).

Neumann elvek

- számítógép működését tárolt program vezérli (Turing);
- a vezérlést vezérlés-folyam (control-flow graph - CFG) segítségével lehet leírni; /lásd vezérlő egység tétel!/ Fontos lépés itt az *adatút* megtervezése.
- a gép belső operatív tárolójában a program utasításai és a végrehajtásukhoz szükséges adatok egyaránt megtalálhatók (**közös utasítás és adattárolás**, a program felülírhatja önmagát – **Neumann architektúra** definíciója);
- az *aritmetikai / és logikai* műveletek (programutasítások) végrehajtását önálló részegység (ALU) végzi; /lásd ALU-s tétel!/. CU – vezérlő egység szeparáció.
- az adatok és programok beolvasására és az eredmények megjelenítésére önálló egységek (IO perifériák) szolgálnak;
- 2-es (bináris) számrendszer alkalmazása.
  - PI: EDVAC computer, ENIAC stb.

Fix vs. **tárolt** programozhatóság

- Korai számítási eszközök **fix** programmal rendelkeztek (nem tárolt programozható): pl: kalkulátor
  - - Program változtatása: „átvezetékezéssel”, struktúra újratervezéssel lehetséges csak
  - - Újraprogramozás: folyamat diagram → előterv spec. (papíron) → részletes mérnöki tervek → nehézkes implementáció
- **Tárolt** programozhatóság ötlete:
  - + Utasítás-készlet architektúra (ISA): RISC, CISC
  - + Változtatható *program*: utasítások sorozata
  - + Nagyfokú flexibilitás, *adatot* hasonló módon tárolni, és kezelni (assembler, compiler, automata prog. eszk.)

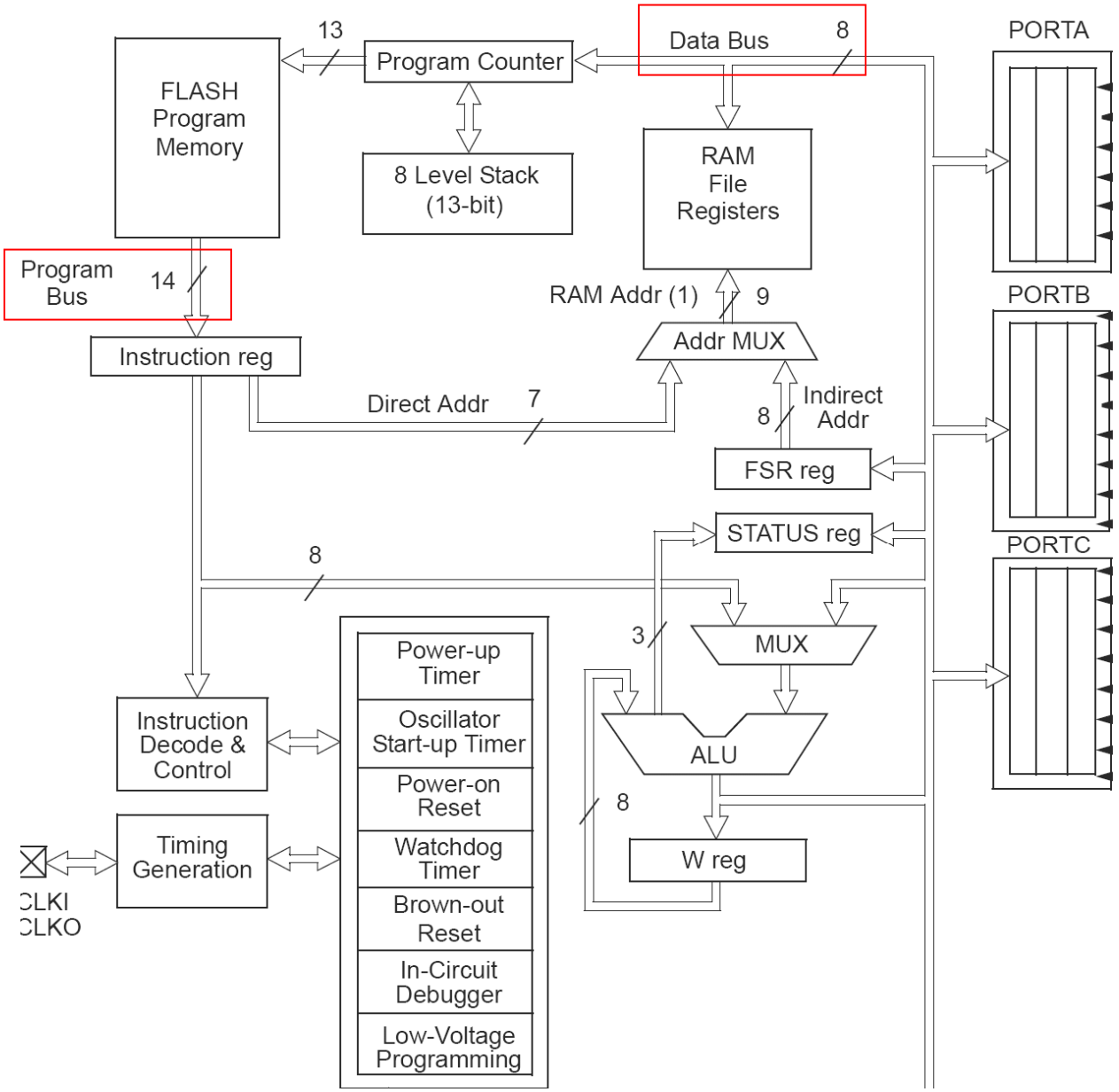
Neumann architektúra hátrányai

- „Önmagát változtató” programok (self-modifying code):
  - Már eleve hibásan megírt program „kárt” okozhat önmagában ill. más programokban is: „malware”=„malfunction”+„sw”.
  - OS szinten: rendszer leállás
  - Pl., Buffer túlcsordulás (overflow): kezelése hozzáféréssel, ill. memória védelemmel
- **Neumann „bottleneck”**: sávszélesség korlát a CPU és memória között (ezért kellett bevezetni a *cache memóriát*), amely probléma a nagymennyiségű adatok továbbítása során lépett fel.
  - Ugyanis a *nem-cache alapú* Neumann rendszerekben, egyszerre vagy csak adat írás/olvasást, vagy csak az utasítás beolvasását lehet elvégezni (egy buszrendszer!)
- A modernebb funkcionális- (XML, SQL, LISP) illetve objektum orientált (pl. C++) programozással már kevesebb adatot mozgatnak meg, mint a korai program nyelvek (pl. Fortran) esetében.
- Neumann bottleneck megkerülésére egy lehetőség a *memrisztor* alapú technológia is.

B.) Harvard architektúra

- Olyan számítógéprendszer, amelynél a program*utasításokat* és az *adatokat* fizikailag **különálló** memóriában tárolják, és külön buszon érhetők el.
  - Eredet: Harvard MARK I. (relés alapú rdsz.)
  - További példák:
    - Intel Pentium processzor család L1- szintű különálló *adat- és utasítás-cache* memóriája.
    - Ti320 DSP jelfeldolgozó processzorok (RAM, ROM memóriái) - / bővebben a DSP-s fóliákon /
    - FPGA alapú beágyazott (embedded) rendszerek: MicroBlaze, PowerPC buszrendszerei, memóriái.
    - Mikrovezérlők (MCU) különálló utasítás-adat buszai és memóriái (PIC-MicroChip, Atmel stb. sorozatok)

Példa: PIC 14-bites mikrovezérlő



Harvard arch. tulajdonságai

- Nem szükséges a memória (shared) osztott jellegének kialakítása:
  - + Szóhosszúság, időzítés, tervezési technológia, memória címzés kialakítása is különböző lehet.
  - Az utasítás (program) memória gyakran szélesebb mint az adat memória (mivel több utasítás memóriára lehet szükség)
  - Utasításokat a legtöbb rendszer esetében ROM-ban tárolják, míg az adatot írható/olvasható memóriában (pl. RAM-ban).
  - + A számítógép különálló buszrendszere segítségével egyidőben akár egy utasítás beolvasását, illetve adat írását/olvasását is el lehet végezni (cache nélkül is).

„Módosított” Harvard architektúra

- Modern számítógép rendszerekben az utasítás-memória és CPU között olyan közvetlen adatút biztosított, amellyel az *utasítás-memóriában lévő szót* is olvasható *adatként* lehet elérni.
  - *Konstans adat* (pl: string, inicializáló érték) utasítás memóriába töltésével a változók számára további helyet spórolunk meg az adatmemóriában
  - Mai modern rendszereknél a Harvard architektúra megnevezés alatt, ezt a módosított változatot értjük.
  - *Gépi* (alacsony) szintű assembly utasítások

Harvard architektúra hátrányai

- Az olyan egychipes rendszereknél (pl. **SoC**: System On a Chip), ahol egyetlen kisméretű chipen van implementálva minden funkció, nehézkes lehet a különböző memória technológiák együttes használata az utasítások és adatok kezelésénél. Ezekben az esetekben a Neumann architektúra alkalmazása lehet megfelelőbb.
- A magas-szintű nyelveket (pl. ANSI C szabvány) nem mindig közvetlenül támogatja (nyelvi konstrukció hiánya az utasítás olvasható adatként való elérésére a külön utasítás memóriából) → assembler szükséges.

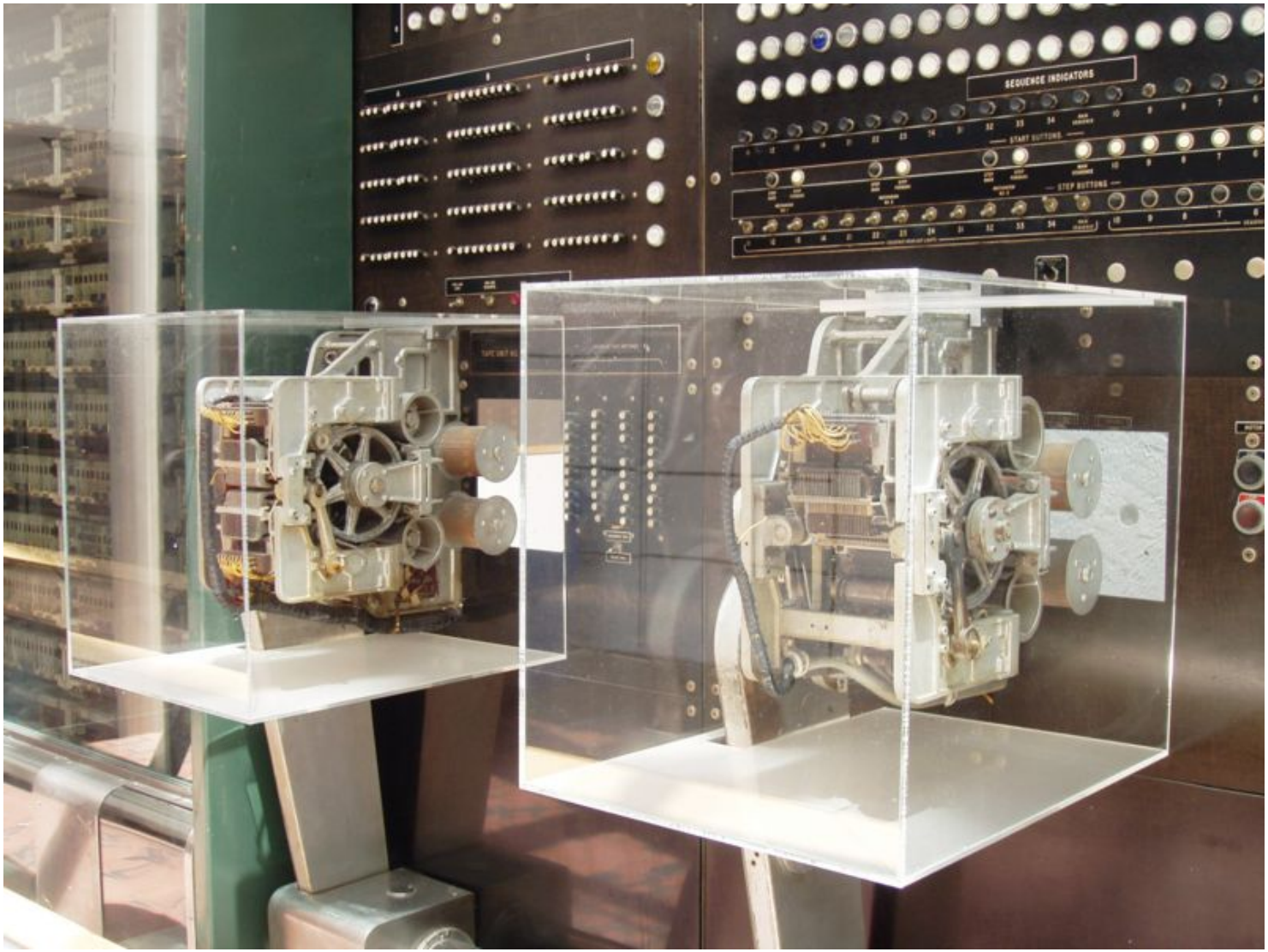
Harvard – Neumann együtt

- Mai, nagy teljesítményű rendszereknél a **kettőt együtt** is lehet, kell alkalmazni.
  - Operatív tár szintjén: von Neumann
  - CPU Cache memória szintjén: Harvard
- Példa: Cache rendszer
  - Programozói szemlélet (Neumann): cache ‘miss’ esetén a fő memóriából kell kivenni az adatot (cím -> adat)
  - Rendszer, hardver szemlélet (Harvard): a CPU on-chip cache memóriája különálló adat- és utasítás cache blokkokból áll.
- *Érdekesség*: vannak olyan számítógép architektúrák, amelyek egyik techológiáról a másikra váltottak időközben: pl ARM 7 (von Neumann) → ARM 9 (Harvard)

Harvard: MARK I

- Howard H. Aiken (Harvard University) – 1944
- Relés alapú aritmetika, mechanikus, korai szgép rendszer. Korlátozott adattároló képesség. (72 db 23 bites decimális számot tárol)
- ***Harvard architektúra***
  - Lyukszalagon tárolt 24-bites utasítások
  - Elektro-mechanikus fogaskerekes számlálókon tárolt 23 bites adatok
  - Utasítást adatként nem lehetett elérni!
- 4KW disszipáció, 4.5 tonna, 765.000 alkatrész: relék, kapcsolók
- Műveletvégzés: +,-: 1 sec, *: 6 sec, /: 15.3 sec
- Logaritmus, trigonometrikus fgv. számítás: 1 min

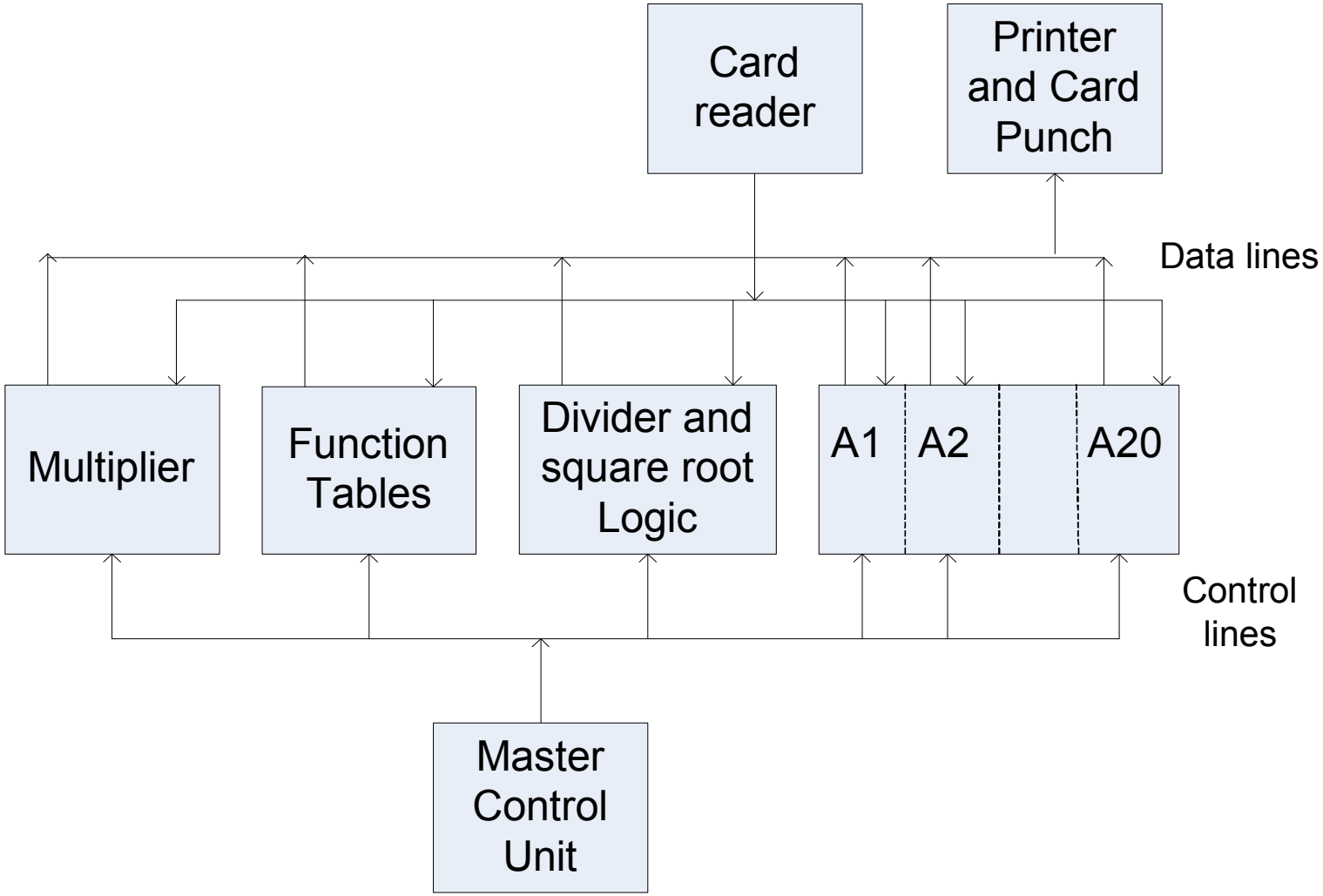
MARK I.



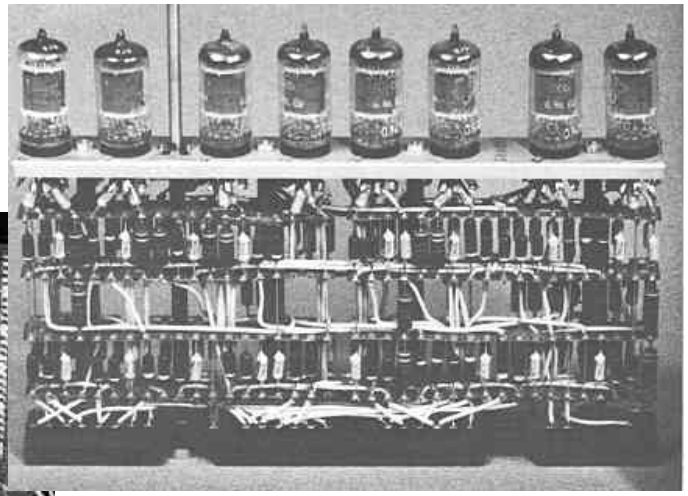
Neumann: ENIAC

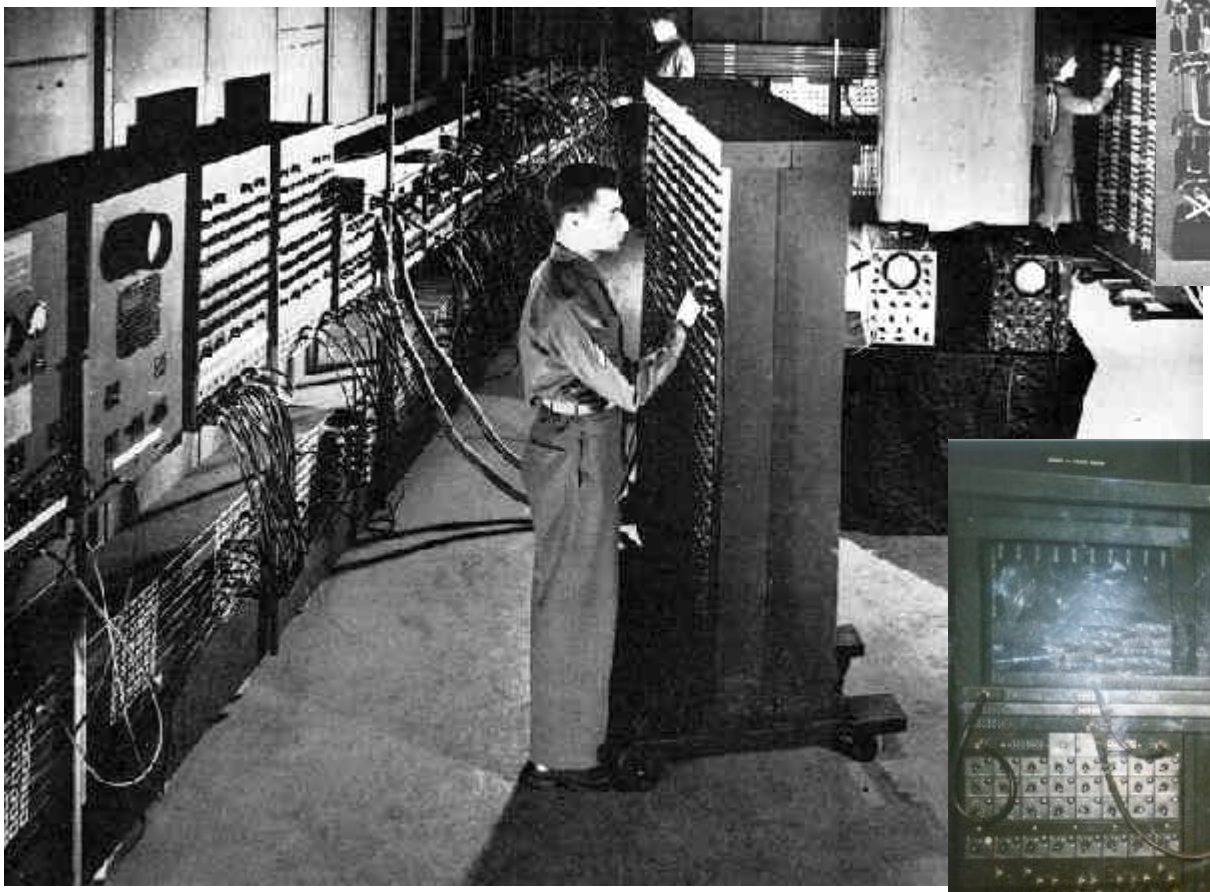
- 1943: **ENIAC**: elektromos numerikus integrátor és kalkulátor (Pennsylvania) Mauchly, Eckert
  - 18000 elektroncső, mechanikus, kapcsolók
  - Gépi szintű programozhatóság, tudományos célokra
  - Összeadás: 3ms
  - 20 ACC reg. – 10 jegyű decimális számra
  - 4 alapművelet + gyökvonás
  - Kártyaolvasó-író
  - Function table: szükséges konstansok tárolása
  - Neumann elvű: közös program/kód és adat

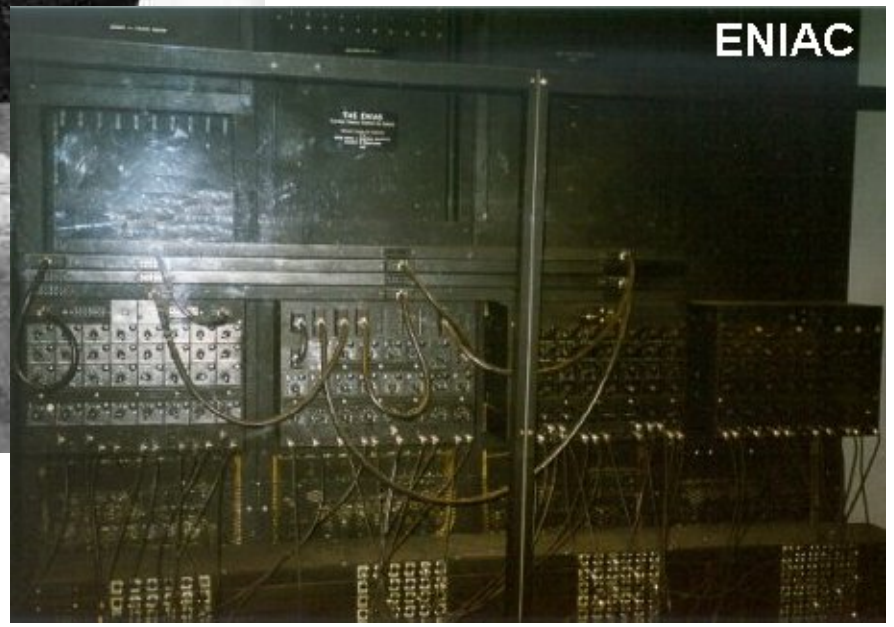
ENIAC


Card reader
Printer and Card Punch
Data lines
Multiplier
Function Tables
Divider and square root Logic
A1
A2
A20
Control lines
Master Control Unit

ENIAC



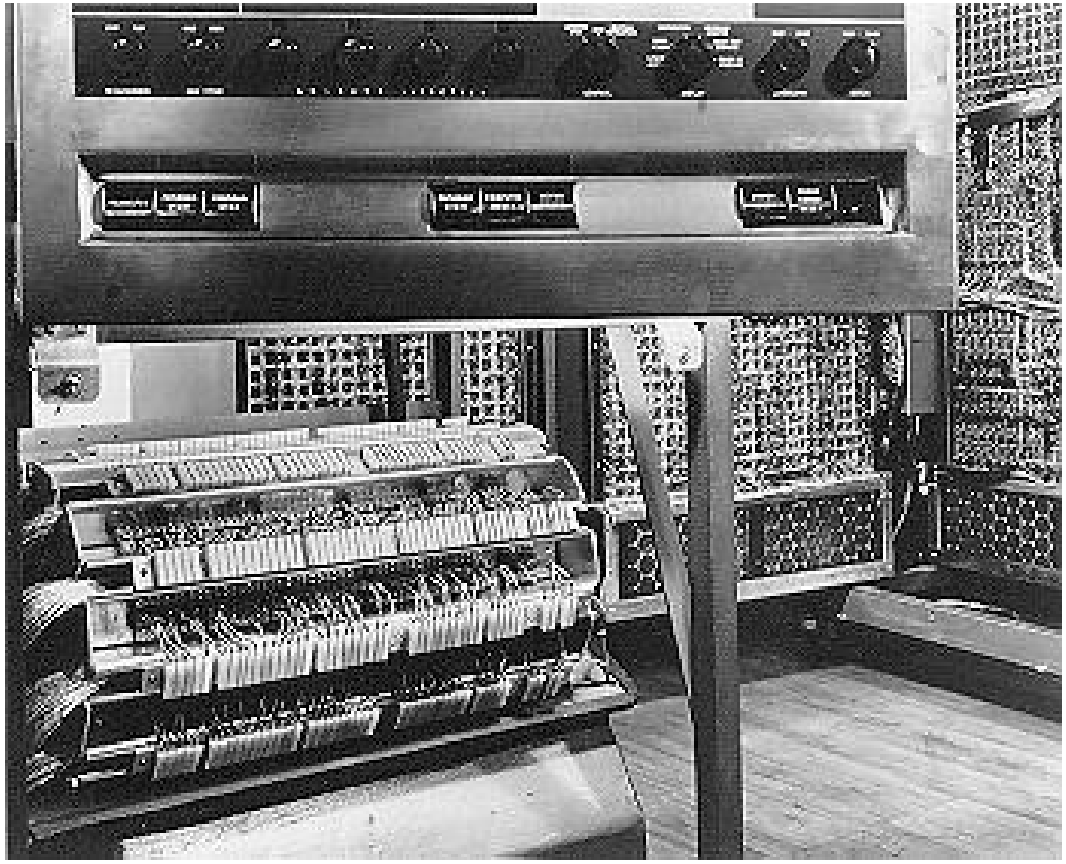




Neumann: EDVAC

- 1945: **EDVAC** (Electronic Discrete Variable Computer): egyenletmegoldó elektromos szgép.
  - Neumann János – „**von Neumann architektúra”**
  - Tárolt programozás
  - 2-es számrendszer
  - 1K elsődleges + 20K másodlagos tároló
  - soros műveletvégzés: ALU
  - utasítások: aritmetikai, i/o, feltételes elágazás
  - EDVAC tanulmány első kivonata [pdf]

EDVAC



Neumann János

Neumann: UNIVAC

- 1951: **UNIVAC** I (UNIVersal Automatic Computer I): üzleti/adminisztratív célokra
  - Mauchly, Eckert tervezte
  - 1951-es népszámlálásra, elnökválasztásra
  - 5200 elektroncső,125KW fogyasztás, 2.25MHz
  - 1000 szavas memória, (12 bites adat: 11 digit + 1 előjelbit, 2x6 bites utasítás formátum)
  - Összeadás: 525μs, szorzás: 2150μs
  - BCD, paritás ell., hiba ell.

UNIVAC - I