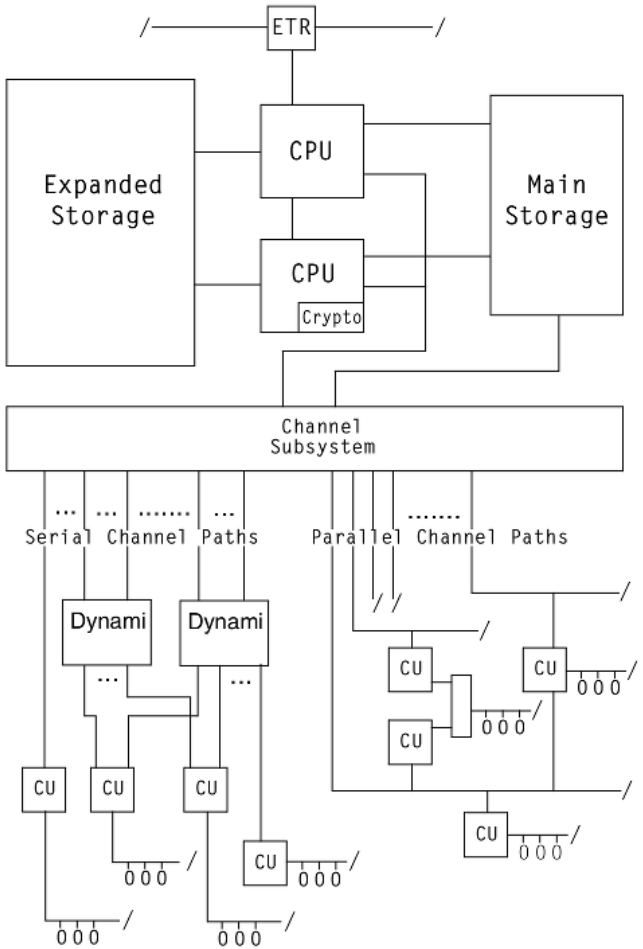
Tomáš Oberhuber

Faculty of Nuclear Sciences and Physical Engineering
Czech Technical University in Prague

Zjednodušené schéma systému z

- základ hardware pro mainframe tvoří:
  - operační paměť - **MAIN / REAL STORAGE**
  - jeden nebo několik CPU
  - několik vstupně výstupních kanálů - **I/O CHANNELS**
  - vstupně výstupní porty - **I/O PORTS**

Zjednodušené schéma systému z



Zdroj: IBM, z/Architecture Principles of Operation

Zjednodušené schéma systému z

- CPU obsahuje cache pro rychlejší práci s daty
- do operační paměti mají přístup všechny procesory a vstupně výstupní kanály
  - systém ošetřuje současný přístup na stejné místo v paměti
  - I/O kanály mají obecně vyšší prioritu než CPU
    - (CPU může počkat, komunikace s okolím ne)
  - priority lze přenastavit

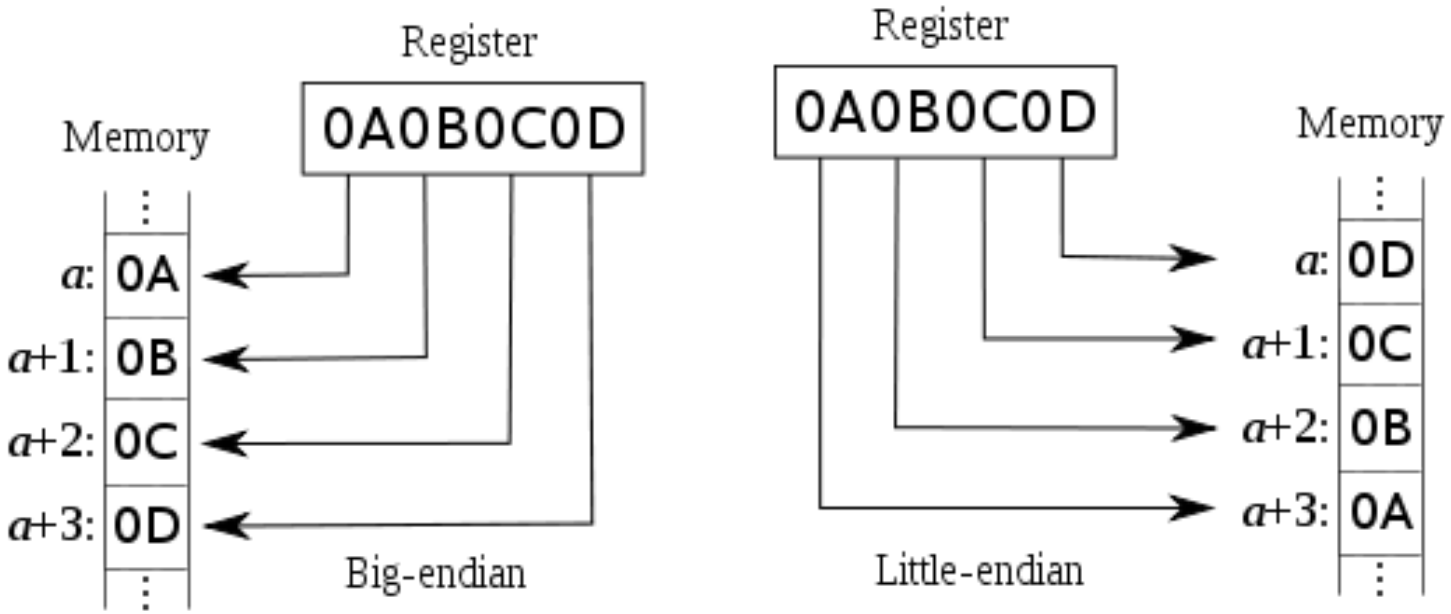
Uložení dat v paměti

- data jsou kódována binárně a skládají se z bitů
- 8 bitů tvoří jeden bajt
  - bity v bajtu číslujeme zleva do doprava od 0 do 7
  - nultý bit je tzv. **high-order bit** a je nejvíce významný (má hodnotu 128 při binárním zápisu celého čísla)
  - sedmý bit je tzv. **low-order bit** a je nejméně významný (má hodnotu 1 při binárním zápisu celého čísla)
- systém obsahuje i bity pro detekci a korekci chyb
  - ECC - error checking and correction
- tyto bity ale nejsou pro programátora přístupné

Uložení dat v paměti

- často potřebujeme pracovat s více bajty současně
  - např. jeden bajt pojme čísla 0 – 255, chceme-li větší, musíme ho zapsat do dvou nebo více bajtů
- kromě bajtu definujeme:
  - **HALFWORD** = 2 bajty
  - **WORD** = 4 bajty
  - **DOUBLE WORD** = 8 bajtů
- bajty opět číslujeme zleva doprava od 0 do 7
- v praxi pak používáme instrukce, které načtou jeden bajt, halfword, word nebo doubleword
- adresa je dána bajtem nejvíce vlevo
- tento bajt je také **nejvíce významný** - tzv. **BIG-ENDIAN**
  - toto je odlišné od PC, které používá **LITTLE-ENDIAN** a první bajt je nejméně významný

Uložení dat v paměti



Mainframe PC

Zdroj: Wikipedie

Uložení dat v paměti

- data musí být v paměti zarovnána (tzv. **aligned**) na **integral boundary**
  - každý typ musí začínat na adrese dělitelné jeho velikostí
    - halfword musí začínat na sudé adrese
    - word musí začínat na adrese dělitelné 4
    - doubleword musí začínat na adrese dělitelné 8
  - v binárním zápisu to lze poznat snadno podle toho, že poslední 1, 2 nebo 3 bity jsou nulové
- od systémy 370 existuje tzv. **byte-oriented-operand feature**
  - umožňuje pracovat s daty na libovolné adrese ovšem za cenu snížení výkonu
- adresy se nakonec překládají pomocí DAT (**dynamic adress translation**) z logických na absolutní (fyzické)

CPU - instrukce

- CPU načítá posloupnost instrukcí z operační paměti a provádí je
- rozlišuje se pět základních skupin instrukcí
  1. **system-control instructions**
  2. **input/output instructions**
  3. **general instructions**
  4. **decimal instructions**
  5. **floating-point instructions**
- system-control a input/ouput instrukce lze provádět jen pokud je CPU v **supervisor state**

CPU - registry

- pro efektivnější zpracování instrukcí obsahuje CPU registry
- jde o:
  1. **general registers**
  2. **floating-point registers**
  3. **control registers**

CPU - registry

- **general registers**
  - je jich celkem 16 a jsou číslované od 0 do 15
  - jejich velikost je 32 nebo 64 bitů
  - lze je použít pro adresování, indexování a základní aritmetické a logické operace
  - při práci s doublewordy je potřeba použít dva registry
    - udá se libovolný sudý registr a k němu se automaticky použije registr o jedna vyšší
    - první registr obsahuje významnější word, druhý méně významný

CPU - registry

- **floating-point registers**
  - jsou 4 a mají velikost 64-bitů
  - jsou indexovány čísly 0, 2, 4 a 6
  - registr může obsahovat hodnotu *float* nebo *double*
    - typ float je vždy uložen v levé tj. významnější půlce
  - pro 128 bitové operandy lze použít dvojice 0 a 2 nebo 4 a 6

CPU - registry

- **control registers**
  - celkem je to 16 registrů po 32 bitech
  - obsahují příznaky různých stavů

I/O kanály

- obstarávají vstupní a výstupní operace
- mohou to být samostatné jednotky nebo mohou být implementovány jako součást CPU
- je definováno rozhraní používané pro komunikaci s těmito kanály

Instrukce

- **zpracování instrukcí**
  - normálně se instrukce zpracovávají sekvenčně
  - zpracování řídí PSW = **program status word**
    - jde o registr obsahující informace nezbytné ke zpracování kódu
  - sekvenční zpracování může změnit:
    - skok – **branch**
    - instrukce LOAD PSW
    - přerušení

Instrukce

- každá instrukce se skládá z:
  - kódu instrukce – **operation code**
  - operandů – **operands**

Operandy

Operandy se dělí na následující:

- registrové – **register operands**
  - může to být odkaz na general, floating-point nebo control register
  - typ registru je udán kódem instrukce
  - číslo registru udává 4bitový tzv. **R-field** nebo může být udáno implicitně v kódu instrukce
- přímé – **immediate operands**
  - je obsažen v instrukci v 8bitovém poli tzv. **I-field**

Operandy

- paměťové – **storage operands**
  - délka operandu může být dána:
    - implicitně tj. kódem instrukce (např. LOAD HALFWORD)
    - explicitně tj. udáním délky pomocí 8bitového pole **L-field**
  - adresa operandu v paměti se udává vzhledem k bázovému registru
    - díky tomu není nutné mít kód a operandy pokaždé na stejné adrese

Adresování

- celková adresa operandu se skládá z:
  - bázové adresy (uložené v obecném registru) - **B** – base
    - číslo registru udává 4bitové pole - **B-field**
    - tomuto registru se pak říká bázový
    - velikost bázové adresy záleží na zvoleném adresování
    - nastavuje se např. na začátek pole/tabulky
  - indexu (uloženém v obecném registru) - **X** – index
    - číslo registru udává 4bitové pole - **X-field**
    - používá se při indexování prvků v poli
  - posunu - **D** – displacement
    - jde o 12bitové číslo uložené v **D-field**
    - lze s ním provádět posun až o 4,096 bajtů
    - používá se pro výběr některé položky ve struktuře

Celková adresa je:

$$Adr = B + X + D$$

Formáty instrukcí

- instrukce má velikost 1–3 halfwordy a musí být v paměti uložená na sudé adrese
- je celkem 8 základních instrukcí:
  - **I** – immediate operand operation
  - **RR** – register-to-register operation
  - **RI** – register-and-immediate operation
  - **RX** – register-and-indexed-storage operation
  - **RS** – register-and-storage operation
  - **SI** – storage-and-immediate operation
  - **SS** – storage-and-storage operation
  - **S** – operation with implicit operand and storage

Kód instrukce

- má délku 1 až 2 bajty
- první dva bity udávají délku a formát

kód	délka	formát
00	1 halfword	RR
01	2 halfwordy	RX
10	2 halfwordy	RS/S/S/RX
11	3 halfwordy	SS

Formáty instrukcí

- **Formát I** – immediate operand
- SVC
  - *supervisor call* – volá službu operačního systému s kódem I_1

Formáty instrukcí

- **Formát RR** – register-to-register operation
- `R1` je číslo registru s prvním operandem
- `R2` je číslo registru s druhým operandem
- `LCR R1, R2`
  - načti hodnotu z registru `R2` do registru `R1` s opačným znaménkem

Formáty instrukcí

- **Formát RI** – register-and-immediate operation
- `R1` je číslo registru s prvním operandem
- `I2` je druhý operand
- `LHI R1, I2`
  - načti halfword se znaménkem do registru `R1`

Formáty instrukcí

- **Formát RS** – register-and-storage operation
- `R1` je číslo registru s prvním operandem
- `B2,D2` je index bázového registru a posunutí
- `R3 (M3)` je index registru s třetím operandem resp. maska
- `IC R1, M3, D2(B2)`
  - načti posloupnost bajtů od adresy `D2(B2)` do registru `R1` v závislosti na 4 bitech masky `M3` – na každý nastavený bit masky se načte jeden bajt, na každý nulový bit zůstane příslušný bajt daného registru nezměněný

Formáty instrukcí

- **Formát RX** – register-and-indexed-storage operation
- `R1` je číslo registru s prvním operandem
- `B2,D2, X2` je index bázového registru a posunutí
- `L R1, D2(X2, B2)`
  - načti fullword z adresy `D2(X2,B2)` do registru `R1`
- `L R1, R3, D2(X2,B2)`
  - do registrů z adresy `R1` až `R3` načti posloupnost bajtů z adresy `D2(B2)`

Formáty instrukcí

- **Formát S** – storage operation
- `B1,D1` je index bázového registru a posunutí
- `TS D1(B1)`
  - nastav CC (condition code) na hodnotu bitu nejvíce vlevo z bajtu na adrese `D1(B1)`, následně zapiš `xFF` na tuto adresu.
  - jde o atomickou instrukci vhodnou pro implementaci zámků

Formáty instrukcí

- **Formát SI** – storage-immediate operation
- `B1,D1` je index bázového registru a posunutí
- `I2` hodnota přímého operandu
- `MVI D1(B1),I2`
  - zapiš bajt `I2` na adresu `D1(B1)`

Formáty instrukcí

- **Formát SS** – storage-and-storage operation
- `B1,D1, L` je index bázového registru, posunutí a délka prvního operandu
- `B2, D2` je index bázového registru a posunutí
- `MVC D1(L,B1),D2(B2)`
  - zkopíruj `L` bajtů z adresy `D2(B2)` na adresu `D1(B1)`
- `AP D1(L1,B1), D2(L2,B2)`
  - přičti číslo ve formátu *packed decimal* o délce `L2` na adrese `D2(B2)` k číslu o délce `L1` na adrese `D1(B1)`