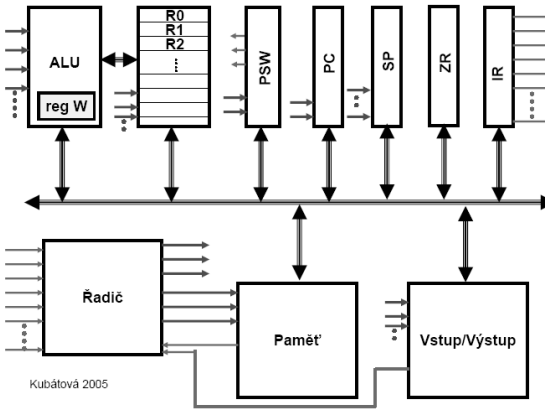
Řadič (Výpisky z přednášky)

Jak to všechno funguje?

Řadič řídí činnost všech výkonných částí počítače podle instrukce.
Řadič je sekvenční obvod
Vstupem jsou stavové signály o stavu procesoru (paměti, sběrnice)
Výstupy jsou řídící signály... říká, co mají ostatní části řadiče potažmo počítače dělat (u každého procesoru to může vypadat jinak)

- write (zapiš něco do nějakého registru)
- write / read - do / z paměti
- oe (output enable) – uvolni data z registru na sběrnici
- register increment (např. pro programm counter)
- register decrement
- HOLD (ještě není hotovo, je třeba ještě čekat)



Z čeho se skládá?

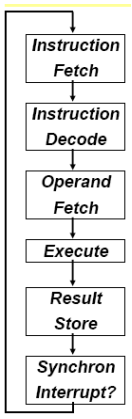
ALU – zpracovává data
W – akumulátor (pro uschování výsledku z ALU)
PC (Program counter) = programový čítač (vysílá na adresovou sběrnici adresy instrukcí, může změnit tu adresu při skoku)
IR (Instruction register)

- pro uchování prováděné instrukce

Stavový registr
ZR (registr obsahující nulu, plánovaný do aDOPu :))
VNITŘNÍ SBĚRNICE (posílání adres, operandů, instrukcí – co tam aktuálně bude bude záležet na řadiči, dobře se to s takovou sběrnicí navrhuje)

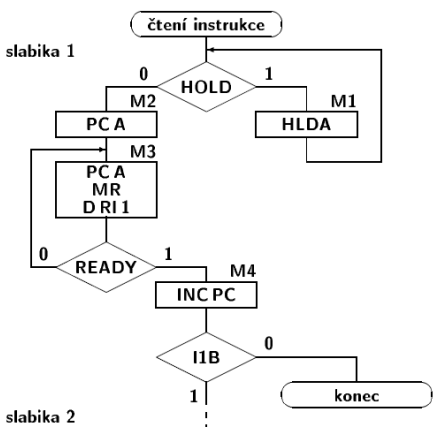
Řadič pracuje v nekonečném cyklu (zpracovává instrukce jednu za druhou)

1. čtení instrukce
2. dekodování instrukce
3. čtení operandu
4. provedení instrukce
5. uložení výsledku
6. eventuelně dotaz na přerušení (pokud je, změní se adresa v programovém čítači)



Ukázka jak pracuje řadič (čtení instrukce):

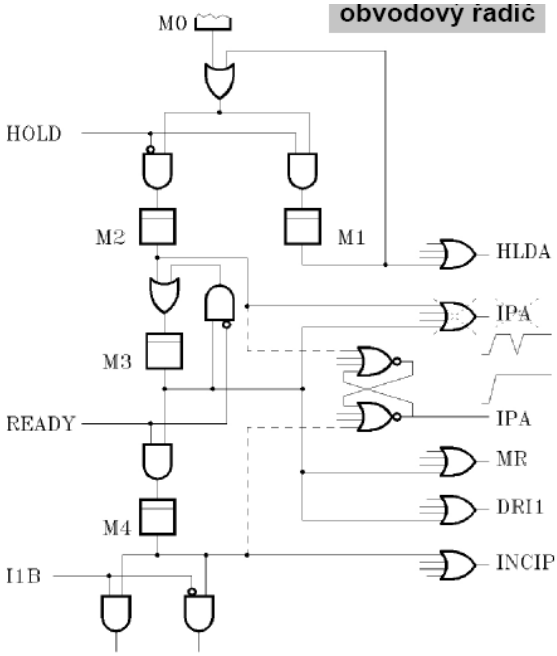
1. žádost o přidělení sběrnice
2. přistoupím na adresovou sběrnici
3. obsah PC hodím na adresovou sběrnici (v tom PC je adresa, odkud chci číst)
4. na adresové sběrnici musí být obsah PC a já budu do cyklu číst z datavé sběrnice, dokud se tam neobjeví moje data
5. jakmile se objeví (přijde signál) musím posunout adresový counter záleží jak daleko se musím dostat (jestli o 16 nebo třeba 32 bitů, udělám PC = PC + 1 nebo PC = PC + 2



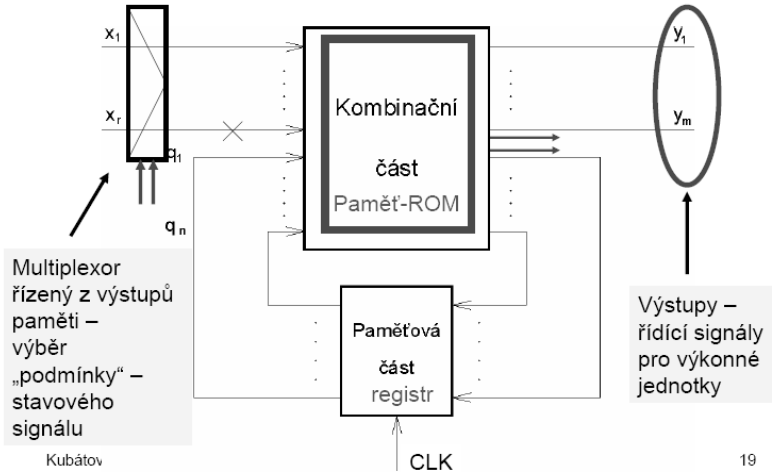
6.

Obvodový (klasický) řadič

- je rychlejší
- nelze v budoucnu měnit



Ilustrace 1: Obvodový řadič



Ilustrace 2: Mikroprogramový řadič (automat)

Mikroprogramový řadič

- dnes obvyklejší (nebo se využívá obou principů)
- lze lépe změnit
- kombinační část je realizována pamětí (všechno, co má řadič dělat je uloženo v paměti)
- mikroinstrukce obsahuje i signály
- zpracování jedné instrukce probíhá v několika taktech (zde analogicky mikroinstrukcích)
- firmware říká, jaké instrukce umí řadič

Pipelining

- urychluje zpracování instrukcí
- dnes již bez pipeliningu neexistuje
- kolize v pipeliningu (triviální řešení = počkat, netriviální = predikovat atd..)

RISC

- redukovaná sada instrukcí (reduced)
- co nejméně instrukcí (aby stačilo co nejméně bitů na zakódování všech instrukcí)
- co nejjednodušší instrukce, aby celá instrukce proběhla co nejrychleji
- co nejjednodušší, aby execute fáze byla co nejrychlejší
- aby šel použít pipelining
- aby šel použít obvodový řaďič .. to je sporné (dnes se na to kouká jinak)
- snaha pracovat s velkým počtem registrů (jsou rychlejší než paměť), komunikace s pamětí bude pouze instrukce STORE nebo LOAD.. to záleží. Jiné instrukce (ADD, SUB, ROL.. všechno bude pracovat jen s registry)

CISC

- rozsáhlá (complex) sada instrukcí
- speciální instrukce pro sčítání 128 bitových čísel, atd (viz semestrálka z NLP)

mnohdy se CISC a RISC trochu prolýnají.. záleží na návrhu