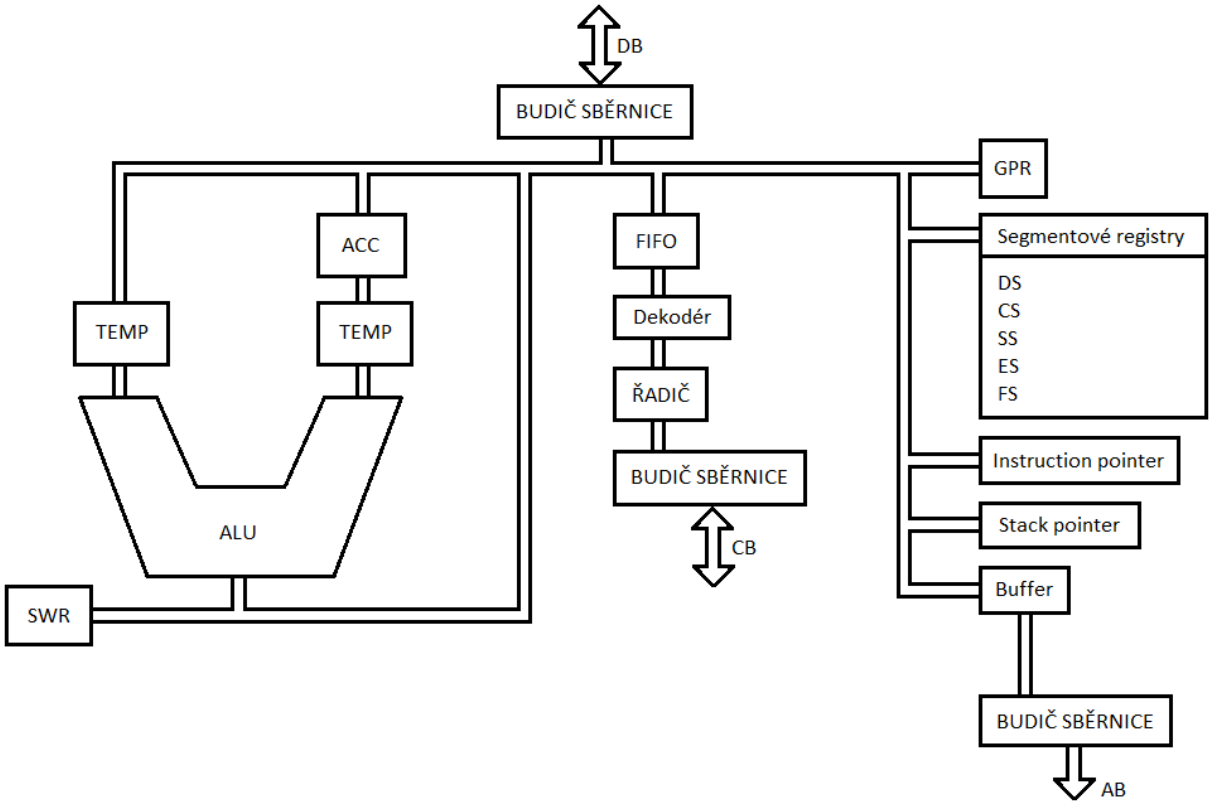
Blokové schéma CPU – součinnost s pamětí při provádění programů (kód, data, stack, podprogram, přerušení apod.), vysvětlit pojmy Harvard vs. von Neumann architektura

Blokové schéma CPU



ACC

Akumulátor nebo také střadač. Registr, v němž je uložen druhý operand při aritmetickologických operacích a kam se po provedení této operace uloží výsledek. Umožňuje tak i postupné přičítání zpracovávaných čísel.

ALU

Část procesoru provádějící samotné výpočty. Prováděné operace jsou na základní úrovni (sčítání, odčítání a základní logické operace). Zpracovává operandy přivedené na vstupy a výsledky operace posílá na výstupu k dalšímu zpracování.

Má za úkol na základě řídících signálů z řadiče provádět matematické a logické operace. Pro práci s reálnými čísly s plovoucí řádovou čárkou je v procesoru integrována výpočetní jednotka FPU – *matematický koprocesor*. ALU a FPU jsou výkonné jednotky.

Budič

Zesiluje signál.

Buffer

Protože se vysílají různé adresy, musí v procesoru být vyrovnávací paměť.

Dekodér

Dekóduje jednotlivé instrukce na mikroinstrukce, podle operačního znaku určí, o jakou instrukci se jedná a podle adresní části, kde se vezme operand.

FIFO

Fronta (FIFO – first in, first out) obsahující 8 položek (instrukcí).

GPR

General propose register – obecně účelné registry. Slouží k tomu, aby se nemuselo komunikovat přímo s pamětí (např. pro mezivýsledky).

Instruction pointer (IP)

Instrukční ukazatel, obsahem je adresa instrukce, která se bude provádět v následujícím taktu. Automaticky se mění, při skoku se *natahuje* nová hodnota.

IP + code segment = adresa instrukce, která se bude provádět v následujícím kroku. Code segment obsahuje číslo segmentu, s nímž se bude pracovat, IP pak offset v rámci segmentu (log. adresu segmentu).

Procesor má buďto právě IP+CS nebo jednoduchý *program counter* obsahující absolutní adresu – používáno zejména u malých procesorů.

IP+CS & program counter – obsah těchto segmentů se vezme a v operační paměti se najde adresa, z buňky s danou adresou se převezme instrukce a začne se provádět. Aby se mohla provádět, musí být její adresa v program counteru nebo v IP+CS. Vzhledem k tomu, že program je sled instrukcí, po přenesení instrukce se program counter automaticky zvedá o jednotku dle délky instrukce (+1 B, +2 B, +4 B, …).

Řadič

Řídicí část – podle dekodéru instrukcí posílá řídicí signály ke všem částem počítače.

Má za úkol číst operandy (data, čísla) a instrukce z operační paměti, dekódovat je a na základě provádění mikrokódu generovat řídící signály (řídit činnost ostatních jednotek v CPU a PC).

Segmentové registry

Hlavně pro vytváření adres.

- **DS (data segment)** – segment dat programu
- **CS (code segment)** – segment kódu programu; nelze přímo číst ani zapisovat
- **SS (stack segment)** – segment zásobníku
- **ES (extra segment)** – řetězce
- **FS (free segment)** – volné použití

Stack pointer

Ukazatel zásobníku – ukazuje na adresu poslední uložené položky (adresy návratové instrukce, lokální proměnné apod.).

Spolu se stack segmentem určují adresu, kam byla naposledy zapsána hodnota. Ukazuje na vrchol zásobníku a ukazuje tedy na doposud nepřečtenou hodnotu. Na této hodnotě je zapsána hodnota stack pointeru a je na ni ukazováno až do doby, než je přečtená. Při novém zápisu se musí nejdříve posunout na novou adresu na nový vrchol zásobníku a teprve poté se může zapsat do SS.

Při čtení ze zásobníku se nejprve přečte hodnota, na kterou ukazuje SP a po přečtení je teprve obsah SP o jednotku směrem dolů změněn (ke dnu zásobníku). Jestliže je něco přečteno, nemůže na takovou věc už SP ukazovat. Dno zásobníku je nejvyšší adresou segmentu.

SWR

Registr stavového slova – drží stav ALU a ACC, dozvídá se, jestli operace (ne)proběhla v pořádku.

Temp

Registry zajišťující konstantní vstup po celou dobu výpočtu, může být připojeno k FPU, slouží k uložení operandu.

Součinnost s pamětí při provádění programů

Pojmy

IP – instruction pointer – ukazuje na adresu v operační paměti, kde je uložena instrukce, která se má v **následujícím** kroku vykonat.

SP – stack pointer – ukazuje do stacku na adresu v operační paměti, kde je uložena instrukce, která by se vykonala, kdyby nebylo vyvoláno přerušení/kdyby nebyl volán podprogram.

Stack – registr obsahující adresy instrukcí, ke kterým se má procesor vrátit po dokončení všech instrukcí z právě vykonávaného (pod)programu či například přerušení.

Kód – **a) strojový kód** (kód na nejnižší úrovni, se kterým pracuje procesor, pro člověka tento kód není srozumitelný; každý procesor má obvykle svoji instrukční sadu); **b) kódy v ostatních jazycích** (pro člověka srozumitelné, aby mohly být zpracovány počítačem, musejí být převedeny na kód strojový prostřednictvím *kompilátoru* – kód je převeden a celý poté spuštěn anebo prostřednictvím *interpreteru* – instrukce po instrukci je převáděna a vykonávána).

Podprogram – je obvykle volán z hlavního programu, vyvolá přerušení vykonávání aktuálně vykonávaného programu, k jemuž se procesor vrátí až po vykonání sledu instrukcí podprogramu.

Přerušení – vyvolané např. pohybem myši, stiskem klávesy nebo z různých důvodů procesorem – stejně jako „odskok do podprogramu“ způsobí přerušení výkonu sledu instrukcí aktuálního programu, k jemuž se procesor vrátí až po vykonání instrukcí související s přerušením, které má v daný moment vyšší prioritu.

Příklad průběhu vykonávání instrukcí včetně vykonání jednoho podprogramu:

1. na začátku ukazuje IP na adresu A0 a SP na adresu D0
2. je načtena instrukce, která se nachází v OP na adrese A0 (*MOV*) a následně je vykonána, IP se zvedne o 1 (jeho nová hodnota je tedy A1 = ukazuje do OP na adresu A1), SP zůstává nezměněn
3. je načtena instrukce, která se nachází v OP na adrese A1 (*INC*) a následně je vykonána, IP se zvedne o 1 (jeho nová hodnota je tedy A2 = ukazuje do OP na adresu A2), SP zůstává nezměněn
4. je načtena instrukce, která se nachází v OP na adrese A2 (*CALL @B0*) a následně by tato instrukce měla být vykonána (má být spuštěna instrukce na adrese B0); IP se opět zvedne o 1 (nová hodnota je A3); procesor uloží adresu instrukce, která by měla být následně vykonána = aktuální hodnotu IP (tedy A3) do stacku a do SP uloží adresu aktuálního vrcholu stacku (adresu CF); do IP je následně uložena adresa volané instrukce (B0)
5. je načtena instrukce, která se nachází v OP na adrese B0 (*ADD*) a následně je vykonána, IP se zvedne o 1 (jeho nová hodnota je tedy B1 = ukazuje do OP na adresu B1), SP zůstává nezměněn (ukazuje stále na vrchol stacku = adresa CF, na které je uložena adresa A3)
6. je načtena instrukce, která se nachází v OP na adrese B1 (*MOV*) a následně je vykonána, IP se zvedne o 1 (jeho nová hodnota je tedy B2 = ukazuje do OP na adresu B2), SP zůstává dále nezměněný
7. je načtena instrukce, která se nachází v OP na adrese B2 (*RET*) a následně je vykonána (RET = return = návrat – k předchozímu programu)
8. vykonávání podprogramu příkazem *RET* skončilo; procesor ze stacku (z adresy, na kterou ukazuje SP) načte poslední uloženou hodnotu (na adrese CF hodnotu A3) a tuto hodnotu uloží do IP (jde totiž o adresu instrukce, která se měla vykonávat, kdyby nebyl zavolán podprogram); v IP je tedy není uloženo A3; SP ukazuje nyní opět na dno stacku
9. je načtena instrukce, která se nachází v OP na adrese A3 (*MOV*) a následně je vykonána, IP se zvedne o 1 (jeho nová hodnota je tedy A4 = ukazuje do OP na adresu A4), SP zůstává nezměněn
10. *pokračování programu*

Procesor

Instrukce	Adr. prov. inst.	IP	SP	Obsah stacku
		A0	D0	AA
MOV	A0	A1	D0	AA
INC	A1	A2	D0	AA
CALL @B0	A2	A3	D0	AA
			CF	A3
		B0	CF	A3
ADD	B0	B1	CF	A3
MOV	B1	B2	CF	A3
RET	B2		CF	A3
		A3	D0	AA
MOV	A3	A4	D0	AA
…	…	…	…	…

OP

Adr.	Hodnota
A0	MOV
A1	INC
A2	CALL @B0
A3	MOV
A4	…
A5	…
…	…
B0	ADD
B1	MOV
B2	RET
…	…

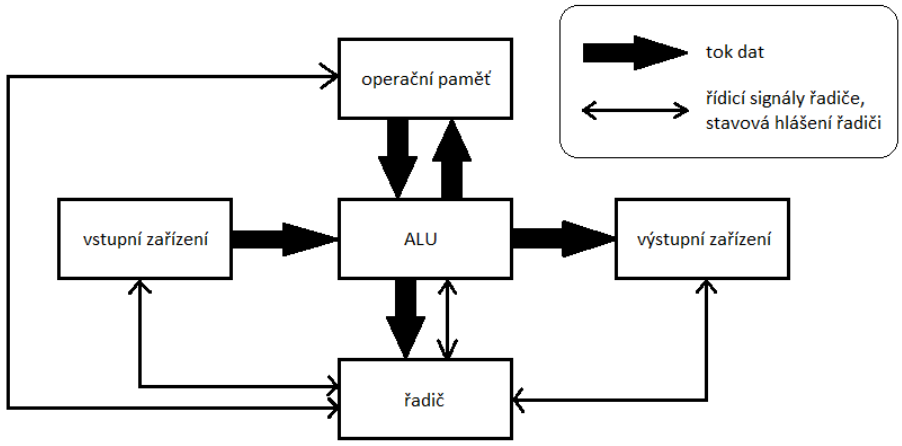
Stack

Adr.	Hodnota
…	…
CB	…
CC	…
CD	…
CE	…
CF	A3
D0	AA

Harvard vs. von Neumannova architektura

Von Neumannova architektura

Principy popsal John von Neumann na své přednášce v Americe. S určitými obměnami je tato struktura základem počítačů dodnes.



Pravidla:

1. počítač se skládá z řadiče, ALU, paměti, vstupních a výstupních zařízení
2. struktura počítače je nezávislá na řešeném problému
3. **v paměti jsou data uložená společně s instrukcemi programu**
4. paměť je rozdělena na buňky stejné velikosti, ke kterým se přistupuje pomocí adresy
5. program je tvořen posloupností instrukcí
6. pořadí provádění instrukcí je sekvenční (postupné) s výjimkou instrukcí skoku
7. instrukce, data a adresy jsou kódovány binárně

Harvardská architektura

Předpokládá existenci dvou oddělených pamětí – první pro instrukce, druhou pro proměnná data. Každá paměť se adresuje samostatně. Umožňuje paralelní čtení instrukce při běhu programu – dochází ke zrychlení zpracování instrukcí.

Tato koncepce se využívá např. v jednoúčelových programovatelných automatech nebo kapesních kalkulátorech.