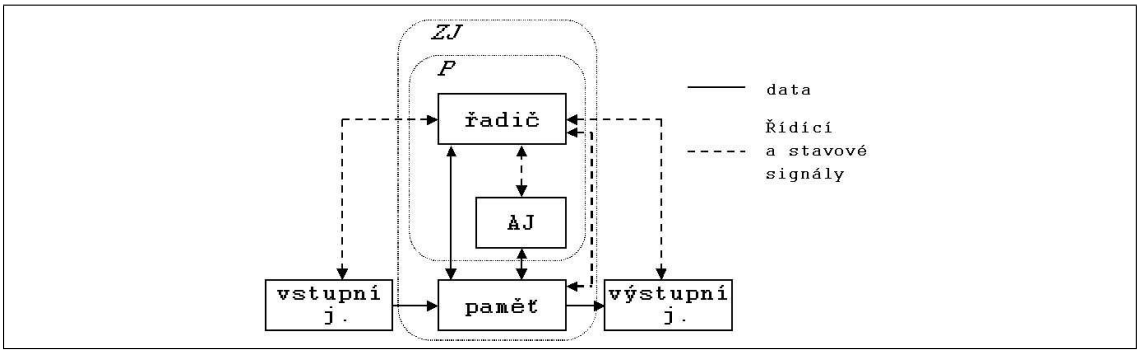
Kapitola 3

Architektury počítačů a procesorů

3.1 Von Neumannova (a harvardská) architektura

Von Neumann

1. počítač se skládá z funkčních jednotek - paměť, řadič, aritmetická jednotka, vstupní a výstupní jednotky - viz. obrázek 3.1
2. struktura počítače je nezávislá na typu řešené úlohy, počítač se programuje obsahem paměti
3. instrukce a data jsou v jedné paměti
4. paměť je rozdělena do buněk stejné velikosti. Jejich pořadová čísla se používají jako adresy. Data ani instrukce nejsou explicitně označeny
5. program je tvořen posloupností elementárních příkazů (instrukcí), v nichž zpravidla není obsažena hodnota operandu (uvádí se pouze jeho adresa) ⇒ program se při změně dat nemění. Instrukce se provádějí jednotlivě v pořadí, v němž jsou zapsány do paměti
6. změna provádění instrukcí se vyvolá instrukcí podmíněného nebo nepodmíněného skoku
7. pro reprezentaci instrukcí i dat se používají dvojkové signály



Obrázek 3.1: Von Neumannova architektura

Harvardská architektura

U této architektury je paměť programů (instrukcí) oddělena od paměti dat.

3.2 Struktura jednoprocesorového počítače

Pozn. Začal bych obrázkem 3.1 Von Neumannova architektura.
Procesor se skládá z AJ, řadiče a registrů.

AJ - aritmeticko logická jednotka

AJ provádí základní logické operace a základní operace s čísly v pevné řádové čárce. Nepracuje s čísly v pohyblivé řádové čárce. Tyto operace zajišťuje buď:

1. jednotka pro práci s čísly v pohyblivé řádové čárce. Může být umístěna přímo v procesoru nebo může být realizována samostatně - koprocesor
2. operace jsou realizovány softwarově programem, tj. jsou převedeny na operace s čísly v pevné řádové čárce a provedeny v AJ

Řadič

Řadič načítá strojové instrukce, dekóduje je a řídí činnost procesoru při jejich provádění. Provedení instrukce spočívá v provedení celé řady dílčích operací (např. zvýšení obsahu programového čítače ...). Tyto dílčí operace se nazývají mikrooperace. Programu, který řídí činnost řadiče se říká mikroprogram.

Registry

Registry slouží pro ukládání mezivýsledků a informací nutných pro řízení činnosti procesoru. Slouží také jako rychlá vyrovnávací paměť.

3.3 Architektura procesorů typu CISC a RISC

3.3.1 CISC - Complex Instruction Set Computer

- počítače se složitým souborem instrukcí
- obsahují složité instrukce, kde její vyvolání způsobí provedení složité operace, nebo sled operací. Složité instrukce se zaváděly z několika důvodů:
  - programy zapsané těmito instrukcemi budou jednoduché, krátké, přehledné a budou obsazovat malý objem paměti
  - programy se budou provádět rychle, protože je třeba číst z paměti menší počet instrukcí
- ke vzniku rozsáhlého souboru instrukcí také přispěla snaha o zachování kompatibility směrem nahoru, tj. každý nový model počítače nějaké instrukce přidá, ale žádnou nezruší
- soubor instrukcí většinou obsahuje více než 200 instrukcí, které se provádějí pomocí mikroprogramu
- instrukce mají proměnlivý formát

3.3.2 RISC - Reduced Instruction Set Computer

- počítače s redukovaným počtem instrukcí
- mají malý počet jednoduchých instrukcí
- instrukce je provedena zpravidla v jednom taktu

- používají klasický, tj. obvodový řadič
- pro spolupráci s hlavní pamětí se používají pouze dvě instrukce - zápis/čtení do/z paměti
- instrukce mají pevnou délku a pevný formát, který přesně vymezuje funkci jednotlivých bitů nebo skupin bitů
- v procesoru se používá velký počet registrů
- složitost se z technického vybavení přesouvá do kompilátoru

3.4 Operační kód, struktura instrukce, základní operace

Instrukce - kódovaný příkaz k provedení operace zapsaný jako číslo. Počet číslic tvořících instrukce se nazývá délka instrukce.
Operační znak - část instrukce, která určuje jaká operace se má provést
Operační kód (instrukční soubor) - různým operacím jsou přiřazeny různé operační znaky
Základní operace - operace, kterým jsou přiřazeny operační znaky. Nejdůležitější *základní operace* jsou aritmetické a logické operace. U aritmetických operací se předpokládá určitá reprezentace čísel, např. dvojková čísla v pevné řádové čárce. Je-li možné použít více reprezentací, obsahuje operační kód odpovídající počet operačních znaků. Charakter výsledků u aritmetických a logických operací se ukládá do *registru příznaků*.

Struktura instrukce

Instrukce obsahuje kromě operačního znaku zpravidla ještě adresní část. Ta určuje, s čím se má daná operace provést. Tvoří ji jedna nebo více adres (vnitřní adresy instrukce).

OZ	a

jednoadresová instrukce

OZ	a_1	a_2

dvouadresová instrukce

OZ	a_1	a_2	a_3

tříadresová instrukce

Jeden počítač může mít několik registrů-střadačů. Bývají organizovány jako adresovatelná paměť ⇒ každému z nich přísluší číslo-adresa. Instrukce pak obsahuje, kromě operačního znaku a adresní části, číslo střadače:

OZ	s	a

Je také možné, že oba operandy jsou uloženy ve střadačích a výsledek se ukládá do střadače, ve kterém byl první operand ⇒ instrukce obsahuje dvě čísla registrů a žádnou adresu, tj. bezadresová instrukce:

OZ	s_1	s_2

Operandy mohou mít pevnou délku nebo mohou být tvořeny různým počtem slabik ⇒ slova proměnné délky.

Je-li délka slova proměnná, musí být nějak určena. Proto se v instrukci určuje nejen adresa první slabiky operandu a_1, resp. a_2, ale také jeho délka l_1, resp. l_2:

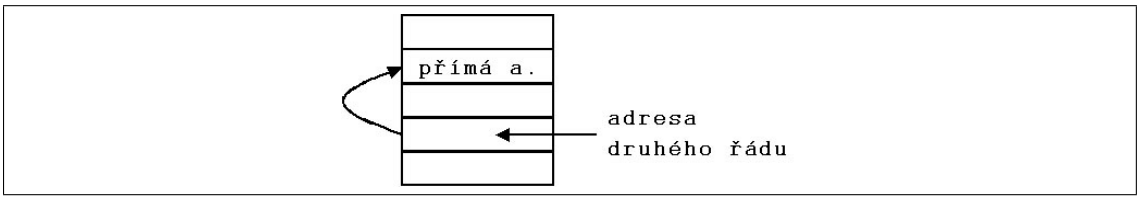
OZ	l_1	l_2	a_1	a_2

Proměnnou délku mohou mít také instrukce, např. v závislosti na počtu vnitřních adres. Délka instrukce je určena operačním znakem.

3.5 Způsoby adresace (typy adres)

Efektivní adresa - adresa, která má být skutečně použita
Přímá adresa - instrukce obsahuje přímo efektivní adresu (operandem, pokud má cenu o něm

mluvit, je její obsah)
Nepřímá adresa - např. adresa druhého řádu - její obsah je efektivní adresou - viz. obrázek 3.2.



Obrázek 3.2: adresa 2. řádu

Aby se poznalo, o který typ adresy jde, používá se v instrukci bit, který se nazývá příznakem nepřímé adresy (0 - jedná se o přímou adresu, 1 - nepřímá adresa). Nepřímé adresy umožňují používat větší adresový prostor než délka adresy instrukce.

Adresa nultého řádu (přímý operand)
- není adresou, ale je přímo operandem. Operaci prováděné s přímým operandem bývá přiřazen jiný operační znak než téže operaci prováděné s operandem v paměti

Indexregistry
- umožňují použití adres jimi modifikovaných. Efektivní adresa se určí tak, že se adresa uvedená v instrukci sečte s obsahem určeného indexregistru. Případné přeplnění se přitom ignoruje. Pro zadání čísla indexregistru je v instrukci vyhrazen odpovídající počet bitů
- jsou určeny pro práci s indexovanými proměnnými (s prvky polí)

Registry báze (bázové registry)
- umožňují použití bázovaných adres (adres modifikovaných registrem báze), které se vyhodnocují stejně jako adresy modifikované indexregistrem
- jsou určeny ke zvětšení adresového prostoru (rozdíl oproti indexregistrům)

Složené adresy (skládáné)
- adresa uvedená v instrukci tvoří nižší řády efektivní adresy, vyšší řády se přebírají z nějakého registru. Tímto registrem bývá odpovídající část programového čítače

Autorelativní (samorelativní) adresy
- efektivní adresa se určí jako součet adresy v instrukci s obsahem programového čítače
- používá se v případech, kdy je žádoucí zkrátit alespoň u některých instrukcí délku adresní části a kdy je pravděpodobné, že efektivní adresa bude dostatečně blízko instrukci

3.6 Přerušení, jejich typy a obsluha, maska přerušení, priority a jejich vyhodnocení

Přerušení - spočívá v tom, že se přestane provádět původní sekvence instrukcí a začne se provádět jiná sekvence, které se říká *rutina přerušení* a začíná na určené adrese - *přerušovací adresa*. Přerušení tedy připomíná skok na přerušovací adresu. Při přerušení se ukládají informace o přerušeném programu a informace, které blíže specifikují přerušení - *data přerušení*. Tyto informace charakterizují stav, v němž se program nacházel před přerušením a dovolují (poté, co je přerušení zpracováno) pokračovat v programu.

Typy přerušení:

Programové přerušení - příčinou přerušení je chyba v programu (přeplnění, dělení nulou apod.)

Technické přerušení - přerušení je způsobené technickou příčinou (výpadek napájení ...)

Přerušení V/V - přerušení způsobená perfierním zařízením. Reprezentují informaci o tom, že nějaké zařízení V/V změnilo podstatným způsobem svůj stav

Vnější přerušení - jsou způsobena příchodem signálů na speciální, tzv. přerušovací vstupy procesoru. Na tyto vstrupy se obvykle přivádějí zvenčí požadavky na komunikaci s jiným zařízením

Instrukční přerušení - přerušení způsobené k tomu určenou instrukcí (INT). Jedná se vlastně o „rafinovanou“ formu volání podprogramu

Současný vznik dvou nebo více příčin přerušení je možný. Příčinám, které mohou vzniknout současně se přidělí různé priority. Vznikne-li současně více příčin přerušení, obslouží se ta, která má větší prioritu.

Jednoúrovňové přerušení - nejde přerušit rutinu přerušení

Víceúrovňové přerušení - opak jednoúrovňového. Zde je třeba zabránit tomu, aby přerušení způsobila příčina, která má nižší prioritu.

Maska přerušení - je registr, kde každému jeho bitu je přiřazena jedna nebo více příčin přerušení. Příčiny přerušení, které jsou přiřazeny nulovým bitům masky jsou zamaskovány. Některé příčiny přerušení nemusí být přiřazeny žádnému bitu masky $\Rightarrow$ jsou stále odmaskovány. Masku přerušení lze nastavovat pomocí příslušné instrukce. Tato instrukce se může použít jako první instrukce rutiny přerušení a zamaskovat tak všechna přerušení.