TECHNICKÉ VYBAVENÍ

ÚVOD DO TEORIE INFORMACE

- objekty okolního světa jsou charakterizovány hmotou a tvarem. Existují dva druhy integrace mezi objekty:

HMOTNÉ – mění se hmota

NEHMOTNÉ – mění se forma

ZÁKLADNÍ SCHÉMA

Informační zdroj – zpráva – přenosový kanál (signál/šum) – příjemce zprávy

ZPRÁVA je uspořádaný soubor znaků, který je sestavován informačním zdrojem. Rozlišitelné prvky se nazývají symboly a jsou relativně nedělitelné. **SYMBOL** je elementární zpráva. Symboly, kterým je přiřazeno grafické znázornění se nazývají znaky. Množina všech využitelných symbolů, ze které vybírá informační zdroj při sestavování zprávy se označuje jako **ABECEDA**. Jestliže je tato množina konečná, hovoříme o diskrétním zdroji zpráv. Pravidla výběru znaků jsou dány syntaxe. Abeceda zdroje spolu se syntaxí tvoří kód informačního zdroje**. KÓDOVÁNÍ** jsou transformační pravidla, podle kterých jsou vysílané znaky přiřazovány vnitřnímu stavu informačního zdroje. Kódování je tedy převedení zprávy do formy signálu. Už samotná znalost kódu představuje určitou zprávu. Zná-li příjemce kód informačního zdroje může usuzovat o jeho vnitřním stavu.

Zpráva je souhrnem určitého množství informace. Znalost, kterou příjemce před přijetím zprávy neměl (a vedla ke změně jeho vnitřního stavu) je definována jako **INFORMACE**. Množství informace ve zprávě je relativní a vždy závisí na příjemci a konkrétní situaci. **SIGNÁL** je hmotný nositel zprávy. **PŘENOSOVÁ CESTA** je fyzikální prostředí, ve kterém se pohybuje signál mezi informačním zdrojem a příjemcem. **PŘENOSOVÝ KANÁL** je soubor technických prostředků nutných k zabezpečení přenosu signálu od zdroje k příjemci.

DATY se rozumí zprávy určené pro strojní zpracování případně výsledek tohoto zpracování. **ÚDAJ** je zpráva získaná jako produkt jistého postupu (výstup měřícího přístroje, senzoru). Převedením údajů do podoby vhodné pro strojní zpracování dostaneme data.

Zpracováním dat, či údajů získáme informace

SYSTEMATIZACE VÝPOČETNÍ TECHNIKY

POČÍTAČ je zařízení schopné přijmout data, aplikovat na ně předepsaný proces a vydat výsledky. Jinými slovy – počítač je elektrický stroj pro zpracování dat (a získávání informací). Data jsou v počítači zobrazena pomocí určité fyzikální veličiny. Nemusí to být nutně elektrické napětí. Tato veličina může mít charakter diskrétní nebo spojitý. Na základě vnitřní reprezentace dat dělíme počítače na číslicové, analogové a hybridní.

Existuje systematizace počítačů podle použití součástkové základny: generace počítačů

- elektronky: 1953 – R4, P1k, štítky, páska, zpožďovací linky, mb buben

- tranzistory: 1959 – R5, P10k, mg páska, feritová jádra

- integrované obvody: 1964 – R6, P1m, mg páska, ferity

- IV. generace – definice sporná

* R – RYCHLOST, P PAMĚŤ

- V. generace – projekt Japonska na začátku 90. let (JIPDEC)

Základní technologie výroby integrovaných obvodů:

- bipolární – nejrozšířenější, jediné napájení, logický zisk, rychlost, cena, TTL

- unipolární – proudový spínač, ECL, CMOS

Problémy spojené se zvyšováním výpočetního výkonu:

stupeň integrace: čas přepnutí logického elementu (což určuje dobu instrukčního cyklu) je limitující
do hodnoty cca 1 nasec

rychlost signálu vodičem: rychlost světla : 0,3m za 1 nasec

počítač nemůže být větší než kopací míč, což přináší další problém – odvod produkovaného tepla

Moore Gordon E. (1964, ředitel Faichild Semiconductor, jeden ze zakladatelů Intelu)

- počet komponent integrovaných na čipu se každých 18 měsíců zdvojnásobí

- počítač slouží k řešení problému reálného světa. Musíme nějak reálný svět do počítače dostat. To je modelování reálného. Dosažením lepších výsledků dosáhneme jak zlepšením modelu, tak zlepšením výpočetního výkonu. Zlepšením modelu je většinou spojeno bezprostředně s pokrokem v ostatních vědních oborech. Zvyšováním početního výkonu při stávající technologii není možné do nekonečna. Reálná jsou omezení daná fyzikálními principy. V budoucnu budou počítače pracovat na jiných principech (chemické, kvantové, elektrické nanopočítače.)

FUZZY LOGIKA

- úhelným kamenem výpočetní techniky je dvoustavová logika. Tvrzení je pravdivé, či nepravdivé. Realita okolního světa je ovšem podstatně složitější. Nic nového – džinisté ve staré Indii používali 7 kategorií pro pravdivost tvrzení:

1, možná je

2, možná není

3, možná je, ale spíše není

4, asi, nelze rozhodnout

5, možná je, ale nejde rozhodnout

6, možná není, ale nelze rozhodnout

7, možná je, možná není a nelze rozhodnout

- tento přístup připouští jak možnost, že o pravdivost tvrzení nelze rozhodnout, tak i možnost, že v naší analýze existují nejistoty. V reálném světě musíme vždy pracovat s nepřesnými údaji (kde je hranice, že je člověk velký a kdy malý??). V klasické teorii množin prvek do množiny patří nebo nepatří. Jiná možnost není. Používání přesných popisů při vágních údajích vede k idealizování skutečného světa a odklonu od reality.

- Fuzzy množina připouští i částečné členství v množině. Prvek do fuzzy množiny patří s určitou pravděpodobností (stupeň příslušnosti, funkce příslušnosti). Takový popis reálného světa je zřetelně dokonalejší. Kardinálním problémem je získání příslušnosti – existuje více možností matematického popisu průběhu stupně příslušnosti.

Jednoduchý příklad: Regulujeme teplotu v místnosti na optimálních 21°C. Konvenční termostat pracuje dvoustavově a v místnosti je buď horko nebo zima

FUZZY REGULÁTOR: 1, Fuzzyaplikace
2, Aplikace pravidel výpočtu
3, Defuzzykace (třeba vážený průměr)

- použití: pračky, metro, výtah, ABS, fotoaparát

- otcem fuzzy logiky je Lotfi A. Zadech. (profesor na University of California)
- prvá práce o fuzzy množinách je z roku 1965
- principy kompatibility: S rostoucí složitostí systému klesá naše schopnost formulovat řešení a významné vlastnosti o jeho chování, až je dosažena hranice, za kterou je přesnost a revelantnost prakticky se vylučující se jevy

HISTORIE VÝPOČETNÍ TECHNIKY – STROJE

1, MARK 1

- spíše než počítač to byl počítací stroj
- postaven v roce 1944 na Hardwarské univerzitě
  (Howard Hathaway Aiken 9.3.1900 – 14. 3. 1973)

- pracoval s čísly o délce 23 míst, prováděl 4 základní operace a měl speciální program na výpočet logaritmu a trigonometrických funkcí
- vstupem byla děrná páska, výstup na děrné štítky
- základem bylo 72 dekadických rotačních čítačů a mechanické relé
- čas pro vynásobení 2 čísel byl 3 – 5 sekund
- původní název ASCC – automatic sequence controller calculator

2, ENIAC

- Electrical numeric integrator and calculator, Pensylvánská univerzita
- používal dekadické desetimístné slovo
- 19000 elektronek, 1500 relé, váha 30 tun, délka 30 metrů, hloubka 1 metr, výška 3 metry
- vstup a výstup děrné štítky
- poprvé použita rychlá paměť: sada registrů s dobou přístupu cca 0,2 msec
- používal hodinové synchronizační impulsy a počítal pulsy
- výpočetní výkon 5000 operací za vteřinu (HP45), spotřeba 175 kw
- konstrukčně 30 nezávislých bloků, 12 akumulátorů (1 akumulátor = sčítačka + střadač)
- Master programmer, který realizoval vložené cykly
- některé výpočty byly prováděný paralelně
- programování se provádělo změnou propojení jednotlivých modulů (6 žen)
- doba práce mezi poruchami byla 5,6 hodiny
- **VYLEPŠENÍ:** motorgenerátor – jednotka pro posuny a 100 slov mag. paměti
  výpočet balistických tabulek – trajektorie 60 sec
  - člověk - 20 hodin
  - analogový Bushův diferenciální analyzátor - 15 minut
  - ENIAC – 30 vteřin

- odpojen ve 23:45 2. října 1955, pracoval celkem 80223 hodiny
- **řešené problémy:** balistika, předpovědi počasí, atomová bomba, kosmické záření, náhodná
  čísla
- původní rozpočet $61 700, konečný 486 804,22$

3, EDVAC, ODVAC

- změnou vnitřní reprezentace čísel na BCD se změnil počet elektronek z 19 000 na 5000.
- výkon 71 000 součtů nebo 1000 násobení za vteřinu

4, M1

- byl jednoúčelovým stroj pro řešení trojrozměrné Fourierovy transformace
- byl používán při výzkumu krystalových struktur
- pracoval v binárním kódu asi 40 operací za vteřinu
- spuštěn do provozu v roce 1952 v ústředním ústavu matematické v oddělení matematických strojů

5, SAPO

- počítač s binární aritmetikou v plovoucí řádové čárce, slovo o délce 32 bitů
- magnetická bubnová paměť o kapacitě 1024 slov, výkon 5 operací za sekundu

HISTORIE VÝPOČETNÍ TECHNIKY – LIDÉ

John Louis von Neumann (1903 – 1957)

- vystudoval chemii v Berlíně a techniku v Curychu
- diplom chem. inženýra v roce 1926
- v červnu 1945 publikoval „First draft of a report to the EDVAC computer“
- fundamentální práce definující principy činnosti počítače

- počítač je tvořen řadičem, ALU, pamětí a obvody vstupu výstupu
- struktura počítače je neměnná a jeho práce je řízena programem uloženým v paměti
- paměť obsahuje jak data tak instrukce
- paměť je tvořena paměťovými místy, do kterých se dá psát nebo z nich číst. Adresa je dána pořadovým číslem tohoto místa
- program je tvořen posloupností instrukcí, které určují elementární změnu stavu počítače
- pořadí prováděných instrukcí je sekvenční, výjimku tvoří skokové instrukce
- instrukce, adresy i data jsou kódovány binárně

Alan Mathison Turing (1912 - 1954)

- britský matematik
- v roce 1936 publikoval článek „On computable numbers“, ve kterém navrhl stroj, dnes známý jako Turingův stroj (konečný automat)
  - testující osoba komunikuje přes počítačový terminál s člověkem a zároveň se systémem umělé inteligence, ale neví, který komunikující je na kterém kanále. Pokud není svými dotazy schopen identifikovat umělý systém, je tento systém inteligentní.

- podílel se (znalostmi) na stavbě COLOSSUS
- Turingův test (1950)
- věřil, že každou činnost lze algoritmizovat

Grace Murray Hooper (1906 – 1992)

- prvá programátorka stroje Navy MARK 1, později UNIVAC
- autorka termínu BUG:MARK 2, 9.9.15:45
- konzultant firmy DEC

Antonín Svoboda

- ČVUT, Př.FUK, 1936 doktor technických věd
- zaměřovač protiletecký – analogový stroj
- 1940 emigroval do USA, prezident ABAT, řád USA
- po válce Strojovka Brno – kalkulační děrovač
- vedoucí laboratoře matematických zdrojů
- 1958 založen VÚMS – vedoucí výzkumu
- projekty počítačů SAPO, EPOS
- 1964 opět emigroval do USA, pracoval u GE
- známé Svobodovi mapy
- přednášel na Kalifornské univerzitě, později profesor
- za zásluhy o rozvoj počítačů, čestný člen IEEE

HISTORIE VÝPOČETNÍ TECHNIKY - PRŮMYSL

- historie výpočetní techniky není historií lidí nebo firem, je to historie strojů
- jediná výjimka je firma IBM (International business machine), založena Hermanem Holleritem (1896 Tatralating machine, 1924 IBM). Rok 1964 byl rokem uvedení do prodeje systému 360. Použití monolitických a hybridních IO – nová architektura a organizace se s malými změnami používala více než 20 let. Podstatou byla stavebnicová struktura, škálovatelnost, jednotný instrukční soubor a struktura dat a unifikovaný způsob připojování periferií. Adresování paměti probíhalo po bajtech, slovo bylo v násobcích bajtů. Instrukční soubor obsahoval 170 instrukcí včetně instrukcí s plovoucí řádovou čárkou. Řídící paměť byla rozdělena na permanentní a semipermanentní, což umožňovalo emulaci jiných počítačů. Adresovací systém pracoval s bázovými registry – snadná relokace programů v paměti. Ochrana paměti byla zajištěna v blocích 2k, chybné čtení vedlo k vyvolání specifického přerušení. Stav stroje byl indikován speciálním stavovým slovem, které obsahovalo údaje o poslední prováděné instrukci. Periferie byly připojené pomocí dvou, autonomně pracujících kanálů – multiplexním (až 256 zařízení) a selektorovém

STROJE: 360/20,30,40,50,60,75,80,95, další řady:370,3090,39

OPERAČNÍ SYSTÉM 360

- umožňoval poprvé v historii běh několika úloh současně
- nejprve pevný, později proměnný počet úloh
- TSO, vzdálené vstupy z terminálů
- programové vybavení pro řízení práce stroje JCL a pro vývoj aplikací: asembler, ALGOL, FORTRAN, COBOL RPG a PC/1

další operační systémy: PCP, MFT, (TSS), VM/370

- také firma DEC (1957) začala v té době vyrábět unifikovanou řadu malých počítačů pod názvem PDP A PDP 11 (1970 programmed dataprocesor, 16 bitů). Konkurence producentům sálových počítačů především cenou (PDP-1 $ 120 000, IBM $ 1 000 000). Byly určeny především k automatizaci a řízení procesů, jak v průmyslu, tak i ve vědě a výzkumu. Úspěšným pokračováním řady PDP 11 byla série VAX (32 bitů). Výroba těchto počítačů byla ukončena zhruba v roce 1985, firma DEC dávno zanikla, ale části počítačů této architektury vyrábí firma MENTEC dodnes!!!!

STROJE: PDP – 11/5,10,20....94

ARCHITEKTURA POČÍTAČE I.

- architektura stoje je globální pohled na všechny podstatné vlastnosti počítače
- původní definice praví, že je to souhrnný přehled množiny registrů, paměti, instrukčního souboru, datových formátů a adresovacích módů
- v podstatě je to popis počítače z hlediska programátora ve strojovém kódu (poprvé definoval Gene Amdahl, hlavní architekt OS 360, IBM)

4 POPISY STROJŮ

- struktura - propojení jednotlivých funkčních bloků
- organizace - dynamická interakce funkčních bloků
- implementace - návrh a obvodová realizace
- funkce - popis stroje jako funkčního celku

ORGANIZACE stroje specifikuje logické seskupení a vzájemné vztahy mezi subsystémy počítače. Je to projekce architektury jednotlivých funkčních jednotek

IMPLEMENTACE stoje je obvodový návrh a realizace počítače pomocí elektronických prvků

OBECNÝ MODUL POČÍTAČE:

- základem logické části počítače tvoří:

- paměť - pasivní zařízení k uložení dat

- procesor - aktivní prvek, který je schopen interpretovat program
- propojení - přenáší data, mění umístění dat nikoli obsah
- transduktor - mění reprezentace dat, jiný způsob kódování

- dále je nutno definovat instrukční soubor, způsoby adresace a prezentace dat

- uvedené skutečnosti popisují sériový neboli skalární počítač: stroj, který postupně (sekvenčně) zpracovává instrukce uložené v paměti. Stejně postupně probíhá zpracování jedné instrukce: čte příkaz, potom operandy, potom vykoná instrukci, potom uloží do paměti výsledky operace

- v architektuře počítačů lze vysledovat dva koncepční přístupy:

- je-li operační paměť společná pro data i programy hovoříme o ***klasické architektuře von Neumannově***

- je-li paměť rozdělená, jedná se o ***architekturu hardvarskou***

ARCHITEKTURA POČÍTAČE II.

OBECNÝ MODEL POČÍTAČE – NĚKTERÉ ZÁKLADNÍ POJMY:

- adresa je označení místa uložení instrukce nebo dat v paměti stroje. je to pořadové číslo udávající pozici adresovatelné jednotky od začátku lineárního prostoru paměti
- instrukce je kódovaný příkaz, který způsobí, že stroj vykoná určitou činnost
- vykonání jedné instrukce se nazývá **instrukčním cyklem**
- vykonání jedné instrukce je složeno z provedení jistého počtu elementárních úkonů
- hodinový cyklus je časový interval mezi dvěma následujícími čely hodinového impulsu – je to nejmenší rozlišitelná časová jednotka instrukčního cyklu
- strojový jazyk je úplný soubor strojových instrukcí (také instrukční sada, soubor)
- počítač je stavový stroj, pracuje synchronně, činnost je řízena hodinovým signálem

- k vykonání instrukce jsou potřebné alespoň tři kroky: vytažení, dekódování a provedení (fetch, decode, execute)

- jsou-li data uložena v paměti potřebujeme ještě další 2 kroky: vytáhnout data a uložit výsledek. Obecně lze provedení instrukce rozbít na 5 logických kroků, což však nic neříká o počtu hodinových textů k provedení instrukce nutných!!!!

SHRNUTÍ

- základními funkčními bloky počítače je paměť, ALU, zařízení vstupu a výstupu a řídící jednotka (řídící logika realizovaná jako řadič)

- řídící jednotka počítače používá dvoustavovou binární logiku reprezentovanou el. napětím (stavy: přítomnost/nepřítomnost el. napětí)

- struktura počítače je neměnná, činnost počítače je řízena programem, což je posloupnost instrukcí stroje. Změny v činnosti stroje je dosaženo změnou programu

- program přestavuje předpis (elementárních úkonů), jak manipulovat s daty, aby byla získána (nová) informace. Skalární počítač používá přímo sekvenční předpis.

- sled instrukcí nemusí být lineární (přeskočení, opakované užití části programu)

- podstatou zpracování dat počítače je aplikace matematických a fyzických operací

- program je uložen v paměti počítače v binární podobě

- data jsou v paměti počítače uložena v binární podobě

INSTRUKCE – ADRESACE

INSTRUKCE:

- co se má udělat, s čím se to má udělat, kam se má uložit výsledek a kde je následující instrukce

- všechny čtyři části instrukce musí být jednoznačně přímo nebo nepřímo určené. Každá instrukce má tedy 4 log. části a 5 polí: operační kód, prvý operand, druhý operand, místo uložení výsledku a adresu následující instrukce.

ČTYŘADRESOVÉ INSTRUKCE:

- obsahují všechny čtyři části instrukce
- historicky se používali u strojů, které jako paměťové médium měly mag. buben
- dnes jsou občas interně používány v některých řídících strukturách procesoru (mikrokódem řízený kontrolér)

TŘÍADRESOVÉ INSTRUKCE:

- neobsahují pole adresy následující instrukce. Ta je určena nepřímo pomocí registru programového čítače (také instrukční registr), který je modifikován prováděnou instrukcí (délkou nebo obsahem) tak, že vždy ukazuje na následující instrukce

DVOUADRESOVÉ INSTRUKCE:

- instrukce nemají přímo vyjádřenou adresu výsledku, předpokládá se, že výsledek bude uložen na místo jednoho z operandů. Dnes nejčastější varianta instrukce

JEDNOADRESOVÉ INSTRUKCE

- předpokládají užití akumulátorů, což je speciální registr, ve kterém je vždy jeden z operandů
a do kterého i bude uložen výsledek prováděné instrukce

BEZ ADRESOVÉ INSTRUKCE

- bez adresové instrukce zásobníkově orientovaných strojů nepotřebují adresy. Výsledek
i operand jsou vždy uloženy v zásobníku. Instrukce naplnění zásobníku (PUSH) a uložení
obsahu zásobníku (POP) však mají adresu. Tyto stroje jsou vhodné jen pro určité výpočty

- programový kód pro tříadresové stroje je nejkompaktnější (nejkratší). Vykonání instrukce je však časově nejnáročnější (dlouhá instrukce – opakované čtení z paměti)
- naopak programový kód pro zásobníkově orientované stroje je nejdelší avšak vykonání instrukce je nejrychlejší (instrukce jsou krátké a většinou není třeba vypočítávat adresy operandů)

- principální výhodou strojů s akumulátorem a zásobníkově orientovaných je fakt, že výsledek operace zůstává v procesoru a může být opakovaně použit v další operaci bez nutnosti čtení či zápisu do paměti. Bohužel akumulátore je jen jeden, což vedlo ke konstrukci strojů s registry pro všeobecné použití. V procesoru je sada registrů (nebo i několik sad), které slouží k uchování mezivýsledků nebo adres o a tudíž eliminují následné přístupy do paměti. Dalším efektem použití registrů je zkrácení délky instrukce.

INSTRUKCE I.

INSTRUKCE:

- je řetězec symbolů, které při interpretaci procesorem způsobí jednoznačnou a definovanou
změnu stavu stroje

- obsahuje:

- operační kód (určuje jaká změna stavu nastane)
- explicitně nebo implicitně zadané identifikátory v adresovém prostoru stroje, které
mají být v dané akci použity jako parametry

- provedení instrukce je jediný způsob jak změnit stav stroje. Vykonaná změna stavu se promítá do dvou oblastí – výsledků operace a nastavení příznaků. Příznak charakterizuje ukončenou instrukci a
může být použit jako vstupní parametr následující operace:

OPCODE	01	Ö2	O3	OX

- instrukci lze také chápat jako datový typ – vykazuje všechny jeho vlastnosti

- instrukční repertoár – množina pravidel integrovaná v hardware procesoru, která zcela přesně určuje jednotlivé stavy stroje (jinak také instrukční soubor)

- instrukční soubor lze dělit podle různých kritérií:

- instrukce privilegované (nejsou dostupné každému programu)
- instrukce neprivilegované (jsou dostupné každému programu)
- instrukce přenosu dat (paměti, registry, zásobníky)
- diadické instrukce (aritmetické a logické operace)
- monodické instrukce (nulování, inkrement, negace, rotace, posuny)
- instrukce větvení, skoků a cyklů
- instrukce volání podprogramů
- instrukce vstupu, výstupu
- instrukce pro řízení stroje

- většina je implementována jako řada podobných instrukcí nebo jako jedna instrukce, která bude mít různé modality (variace operačního kódu)

INSTRUKCE – ADRESACE OPERANDŮ

- způsob adresace: určuje jak bude k datům nebo instrukcím přistupováno
- adresa uvedená v instrukci často není adresou finální (neukazuje přímo na místo uložení)
- stroj musí vykonat jistý postup (předepsaný v instrukci), aby získal skutečnou, konečnou adresu – efektivní adresu

- pro existenci různých způsobů adresace je nejméně:

1 dobrý důvod
- čím variabilnější jsou módy operace, tím elegantnější a efektivnější by měl být programový kód

1 špatný důvod
- spletitý postup získání konečné adresy komplikuje konstrukci procesoru a prodlužuje dobu potřebnou pro vykonání instrukce)

BEZPROSTŘEDNÍ ADRESACE:
Instrukce obsahuje operand. Data jsou přímo součástí instrukce. Vložené hodnoty jsou neměnné (konstantní), proto hovoříme o instrukcích pro práci s literálem (literál je datová položka, která nemá vlastní identifikátor)

PŘÍMÁ ADRESACE:
instrukce obsahuje adresu operandu. Tato adresa je adresou efektivní

NEPŘÍMÁ ADRESACE:
Instrukce obsahuje adresu, na které je adresa operandu

REGISTROVÁ PŘÍMÁ ADRESACE:
Instrukce obsahují číslo (malou adresu) registru, který obsahoval operand. Velmi efektivní instrukce – doba potřebná k vykonání je zvláště krátká

REGISTROVÁ NEPŘÍMÁ ADRESACE:

Instrukce obsahuje číslo registru, ve kterém je uložena adresa operandu. Tento adresový mód je obzvláště efektivní, potřebujeme-li adresu operandu plynule měnit (práce s bloky dat). Čehož možno dosáhnout prostým přičítáním nebo odečítáním k registru, ve kterém je uložena adresa (a také ji příslušně modifikovat)

REGISTROVÁ NEPŘÍMÁ ADRESACE S POSUNEM:
Instrukce obsahuje číslo registru, ve kterém je uložena adresa a posun. Efektivní adresa operandu se získá sečtením těchto dvou hodnot. Posun musí být konstanta nebo číslo registru (proměnná hodnota uložená v registru)

- při adresování instrukcí se používá relativní adresace (vzhledem k programovému čítači). Instrukce pro řízení chodu programu (například podmíněný a nepodmíněný skok, volání programů) obsahují konstantu („vzdálenost místa skoku“), jejímž přičtením k programovému čítači získáme adresu následující instrukce (je to jistá modifikace bezprostřední adresace). POZOR, konstanta neudává, kolik, instrukcí se má přeskočit, ale kolik nejmenších adresovatelných jednotek (většinou bajtů se přeskočí)!!!! Velikost pro konstantu (počet bitů) limituje minimální délka skoku (1 bajt $\approx$ skok $\pm$ 127 bajtů)

INSTRUKCE II.

ARCHITEKTURA RISC a CISC

RISC – reduced instruction set computer
CISC – complex instruction set computer

RISC:

- minimální instrukční soubor
- jednoduché způsoby adresování
- pevný formát instrukce
- vykonání instrukce v jednom strojovém cyklu
- datové operace pouze nad registry
- styk s paměti výlučně instrukcemi load/store
- zřetězená realizace instrukcí
- nutností je optimalizující kompilátor

ETAPY VÝVOJE RISC:

1. etapa (1975 - 1982)

- pouze expocimentální stroje, realizován pouze procesor
- procesor IBM 801

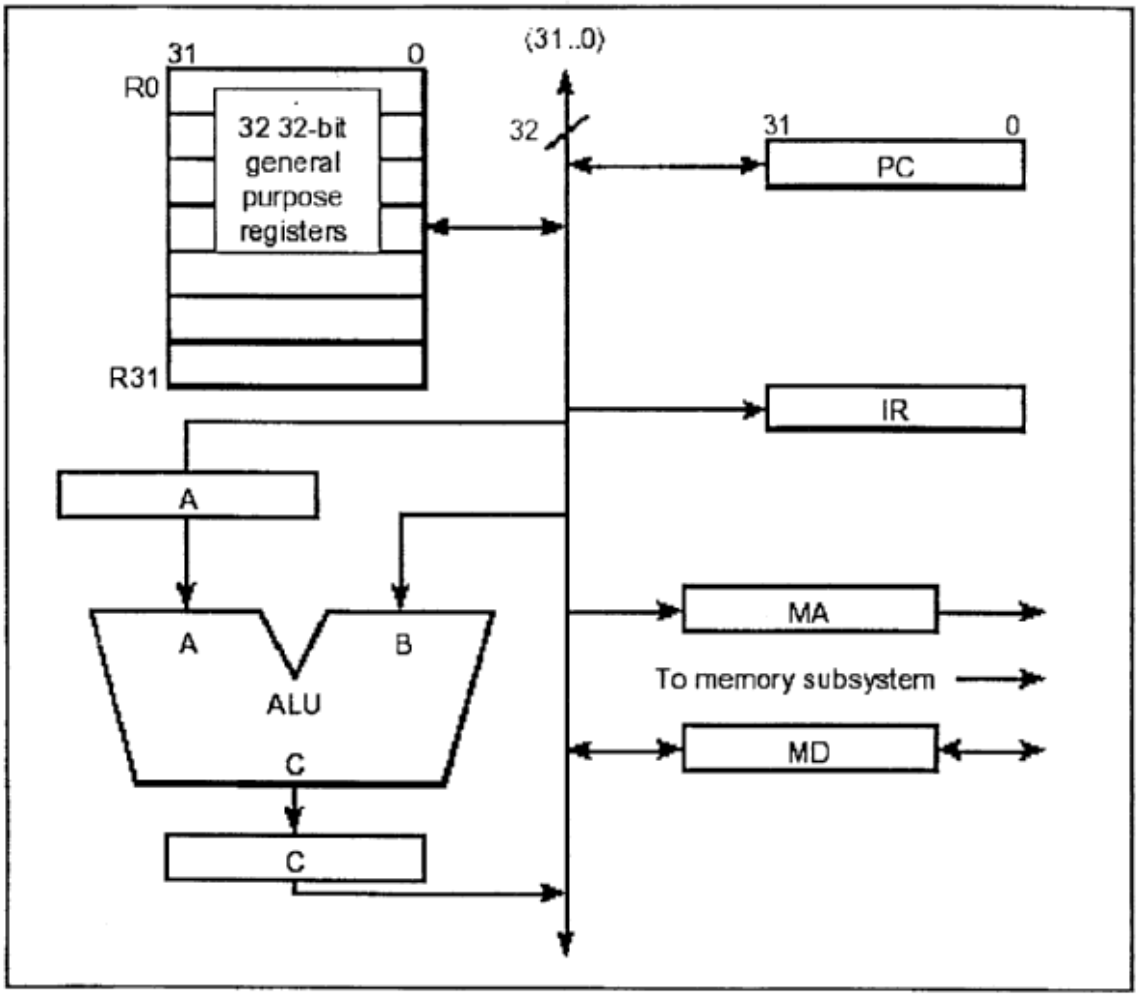
2. etapa (1982 – 1985)

- 1. generaci prakticky použitelných strojů

3. etapa (1985 -)

- Seymour Cray (CDC)
- John Cocke (IBM)
- David Pattersen (Berkeley)
- John Menessy
- Sun SPARC
- Acorn ARM
- Motorola 88000
- DEC APLHA21064A

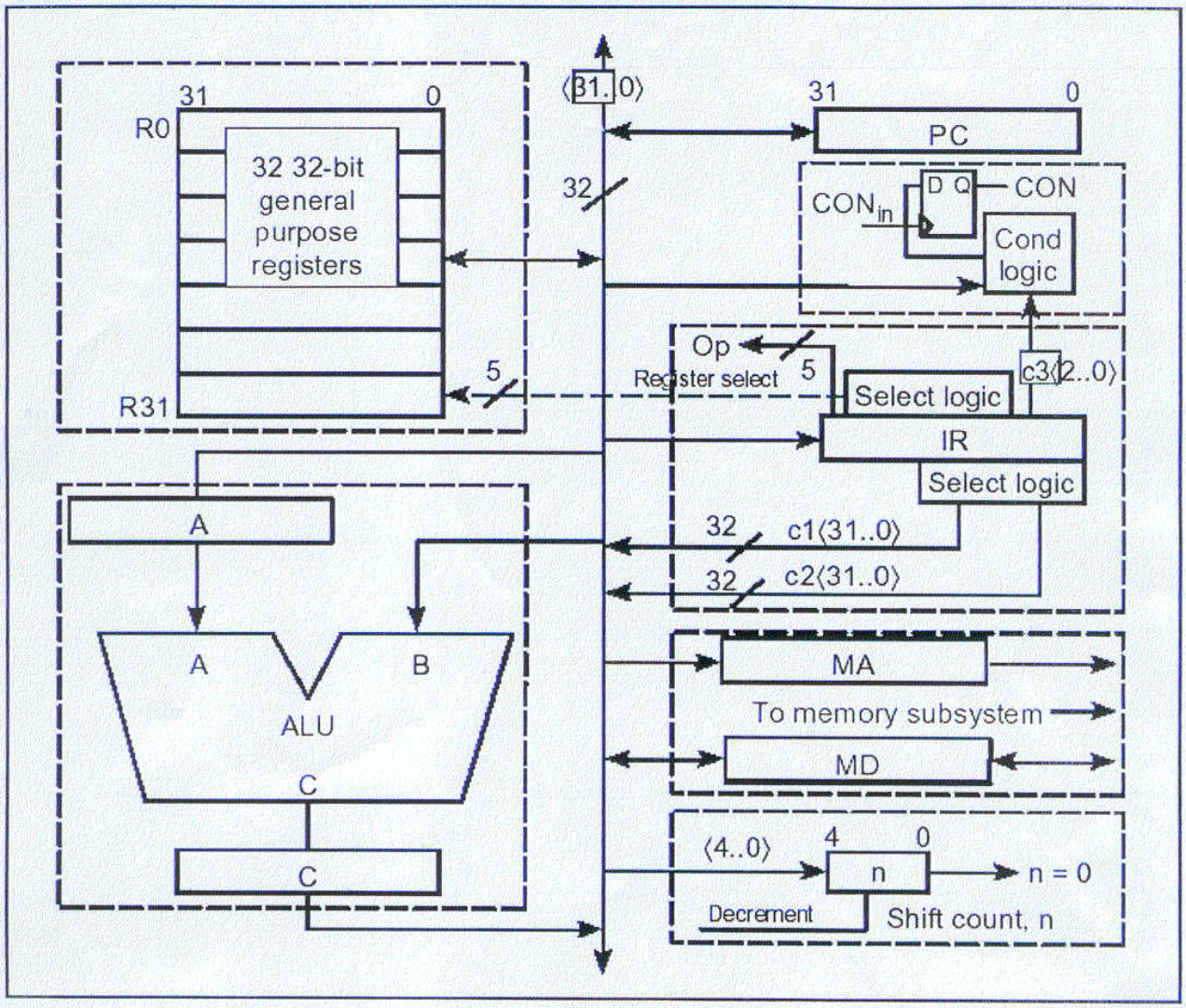
PROCESORY.



INSTRUKCE ADD, rA, rB, rC	
KROK	POPIS ČINNOSTI
T0	MA ← PC : PC + 4
T1	MD ← M (MA) : PC ← C
T2	IR ← MD
T3	A ← R (rB)
T4	C ← A + R (rC)
T5	R (rA) ← C

PROCESORY IV.

Logická struktura procesoru.



Logická struktura procesoru je složena z bloků :

1. Soubor 32 dvaatřicetibitových registrů
2. Aritmeticko logická jednotka
3. Realizace podmíněných skoku
4. Dekódování instrukce a extrakce konstant
5. Rozhraní pro styk s operační pamětí
6. Čítač posuvů

PŘEHLED INSTRUKCÍ.

Nr.	Binárně	Mnemonika	Slovní popis	Instrukce	
0	0 0000	nop	Žádná operace		
1	0 0001	ld	Naplnit registr	ld	rA K
2	0 0010	ldr	Naplnit registr relativně	ldr	rA, D
3	0 0011	st	Uložit obsah registru	st	rA, K
4	0 0100	str	Uložit obsah registru relativně	str	rA, D
5	0 0101	la	Naplnit registr adresou	la	rA, K
6	0 0110	lar	Naplnit registr adresou relativní	lar	rA, D
7	0 0111				
8	0 1000	br	Předání řízení na návěští	br	rB, rC, P
9	0 1001	brl	Předání řízení s návratem	brl	rA, rB, rC, P
10	0 1010	een	Povolit přerušení		
11	0 1011	edi	Zakázat přerušení		
12	0 1100	add	Sčítání	add	rA, rB, rC
13	0 1101	addi	Přičtení konstanty	addi	rA, rB, K
14	0 1110	sub	Odčítání	sub	rA, rB, rC
15	0 1111	neg	Druhý dvojkový doplněk	neg	rA, rC
16	1 0000	svi	Uložit kontext při přerušení	svi	rA, rB
17	1 0001	ri	Vrátit kontext	ri	rA, rB
18	1 0010				
19	1 0011				
20	1 0100	and	Logický součin	and	rA, rB, rC
21	1 0101	andi	Logický součin s konstantou	andi	rA, rB, K
22	1 0110	or	Logický součet	or	rA, rB, rC
23	1 0111	ori	Logický součet s konstantou	ori	rA, rB, K
24	1 1000	not	Prvý dvojkový doplněk	not	rA, rC
25	1 1001				
26	1 1010	shr	Posun obsahu registru doprava	shr	rA, rB, P
27	1 1011	shra	Aritmetický posun reg. doprava	shra	rA, rB, P
28	1 1100	shl	Posun obsahu registru doleva	shl	rA, rB, P
29	1 1101	shc	Rotace obsahu registru doleva	shc	rA, rB, P
30	1 1110	rfi	Návrat z přerušení		
31	1 1111	stop	Zastavit stroj		

PROCESORY A INSTRUKCE

EFEKTIVNÍ ADRESA:

- počítána nebo platná po vykonání instrukce
- máme definovány dva způsoby adresace operandů – absolutní: umístění je dáno hodnotou K (bezprostřední adresace, když rB =0 nebo bázovaná, když rB ≠ 0) a relativní, když je výsledné umístění dáno jako součet programového čítače a konstanty K

INSTRUKCE PŘENOSU DAT:

- zajišťují přemísťování operandů (data nebo adresy). Žádná nová informace nevzniká. Základem jsou instrukce LOAD (přemístění dat do procesoru) a STORE (uložení dat z procesoru do paměti)

KROK	INSTRUKCE LOAD	INSTRUKCE STORE
T0-T2	Vytažení instrukce	Uložení instrukce
T3	A ← ((rB=0) → 0 : (rB≠ 0) → R[rB]	A ← ((rB=0) → 0 : (rB≠ 0) → R[rB]
T4	C ← A+K[IR]	C ← A+K[IR]
T5	MA ← C	MA ← C
T6	MD ← M[MA]	MD ← R[rA]
T7	R[rA] ← MD	M[MA] ← MD

- instrukce mají celkem tři varianty: absolutní nebo relativní adresu pro data a přesun adresy

PROCESORY A INSTRUKCE I.

aritmetické a logické instrukce

– pomocí této skupiny instrukcí vzniká v procesoru nová informace. Aritmetické operace se provádějí s operandy numerického charakteru, charakter operandů logických operací je většinou nenumerický (práce se symboly, zpracování grafiky) V této skupině jsou jak instrukce monodické (negace) tak diadické.

Instrukce addi rA, rB, K (bezprostřední adresace)

KROK	POPIS ČINNOSTI
T0-T2	Vytažení instrukce
T3	A ← R[rB]
T4	C ← A+K
T5	R[rA] ← C

Instrukce neg rA, rC (monodická instrukce)

KROK	POPIS ČINNOSTI
T0-T2	Vytažení instrukce

T3	C ← -R[rC]
T4	R[rA] ← C

instrukce shr rA, rB, P nebo shr rA, rB, rC

KROK	POPIS ČINNOSTI
T0-T2	Vytažení instrukce
T3	n ← P
T4	n = 0 → (n ← R[rC])
T5	C ← R [rB]
T6	Opakuj krok T5 n krát
T7	R[rA] ← 0

- provedení kroku T6 závisí na variantě instrukce posuvu (formální popis je poměrně složitý)

INSTRUKCE ŘÍZENÍ CHODU PROGRAMU

- slouží k měnění posloupnosti vykonávaných příkazů

- žádná nová informace nevzniká

- podle provedení je dělíme na:
  a, instrukce bez návratu (podmíněné a nepodmíněné skoky)
  b, instrukce s návratem (volání podprogramů)

- instrukce nepodmíněného skoku předá řízení jiné části programového kódu (je to podmíněný skok, jehož podmínka je splněna vždy)

- skupina instrukcí podmíněných skoků potřebuje k rozhodnutí, zda se krok provede splnění určité podmínky

- někdy je žádoucí, aby řízení programu vrátilo po vykonání požadovaného úkonu zpět a původní činnost mohla pokračovat. Předpokladem je zapamatování návratové adres (adresy instrukce následující za skokem) a nějaký mechanizmus (instrukce návratu), který obnoví chod původní větve.

instrukce br rB, rC, P {2..0}

KROK	POPIS ČINNOSTI
T0-T2	Vytažení instrukce
T3	CON ← cond (R[rC])
T4	COND → PC ← R[rB])

instrukce brl rA, rB, rC, P {2..0}

KROK	POPIS ČINNOSTI
T0-T2	Vytažení instrukce
T3	R[rA] ← PC
T4	CON ← cond (R[rC])
T5	COND → PC ←R[rB])

P {2..0}	***POPIS PODMÍNKY***	***MNEMONIKA***	
000	neskočí nikdy	brnv	brlnv
001	skočí vždy	br	brl
010	skočí, když r[rc] = 0	brzr	brlzr
011	skočí, když r[rc] ≠ 0	brnz	brlnt
100	skočí, když r[rc] ≥ 0	brpl	brlpl
101	skočí, když r[rc] < 0	brmi	brlmi

ASEMBLERY A ZAVADĚČE

STROJOVÝ KÓD (MACHINE CODE)

- je programovací jazyk nejnižší úrovně, jehož instrukce je schopen procesor bezprostředně vykonávat. Představuje binární zápis v němž se počítačové instrukce ukládají do paměti pro přímé použití procesorem počítač (člověku prakticky nesrozumitelný)

ASEMBLER

- je sestavující program (překladače) jazyka symbolických instrukcí do strojového kódu nebo do přemístitelného kódu (relocatable code). Jedna instrukce v asembleru se převede na jednu instrukci v strojovém kódu

JAZYK SYMBOLICKÝCH ADRES

- je strojově orientovaný jazyk, jehož instrukce nebo jejich části se mohou zapisovat symbolicky. Představuje notaci pro pohodlnější reprezentaci programů ve strojovém kódu v symbolice, která je (člověku) lépe čitelná

- sestavení může být absolutní nebo relativní (výsledkem je kód s relativními adresami vzhledem k začátku programové sekce). Je – li programovacích sekcí více nebo jsou – li užity moduly z knihovny následuje spojení do jednoho souboru, což provede sestavovací program

SESTAVOVACÍ PROGRAM

- je program sloužící k vytvoření spustitelného (zaveditelného) modulu
- řeší 3 základní úlohy:

1, spojení jednotlivých programovacích sekcí a modulů
2, vyřešení vnitřních odkazů mezi nimi
3, konečné nastavení adres

- následuje zavedení do paměti (a vykonání)
- poslední 2 kroky lze spojit do jednoho pomocí programu zvaný zavaděč (loader), varianta závisí na operačním systému

Asembler. Vstupem asembleru je text. soubor (zdrojový text), který má určitou strukturu. Každá řádka musí mít pole návěští, operace, operandy a komentář. Pole může být prázdné, ale musí existovat. Pole jsou oddělené oddělovacími znaky (tabulátor, mezera). Zdrojový text obsahuje instrukce, data a pseudoinstrukce (instrukce pro řízení činnosti asembleru). Asembler převádí symbolické instrukce a adresy na strojový kód. Podstatou úkolu je vyhledání vzorů (symboly instrukcí a adres) a jejich následná transformace na numerické (binární) hodnoty. Používá k tomu tabulku symbolů a čítače polohy (cosi jako programový čítač). Tabulka symbolů shromažďuje relace, symbol, typ a hodnotu. Problémy způsobují dvě skutečnosti: symbol ještě není v tabulce a případně proměnná délka instrukce.
Asembler zpracovává zdrojový text v několika průchodech. První průchod prověří formální správnost a vytvoří tabulku symbolů, druhý průchod provede transformaci, inicializaci proměnných a přiřadí informaci pro spojovací program.

PROPOJOVACÍ SUBSYSTÉMY

- propojovací subsystémy můžeme rozdělit na vnitřní a vnější. Každé propojení je charakterizováno rychlostí, šířkou pásma a pořizovací cenou

- propojovací systém je většinou hierarchický. Význačnou vlastností je způsob šíření informace mezi jednotlivými jednotkami – organizace subsystému

- směrová – vždy bude aktivována zvláštní cesta (část subsystému e neúčastní)

- sběrnicová – informace jsou předávány po obecných cestách

- sériový kanál – informace je přenášena po elementárních cestách

- paralelní kanál – přenášejí se větší jednotky informace

SMĚROVÁ ORGANIZACE

- propojovací síť je složena z uzlů, které komunikují prostřednictvím spojů. Každý spoj je dvoubodové spojení – na jednom spoji se nemůže podílet více uzlů. Způsob propojení jednotlivých jednotek sítě se nazývá TOPOLOGIE propojovacího systému. Vlastnosti navržené topologie určují, jakým způsobem budou jednotky sdílet zdroje a co toto sdílení bude stát. Nejjednodušší topologie je propojení dvou jednotek (bod – bod). Komunikační strategie je algoritmus směrování zpráv na cestě od uzlu k síti

PROPOJOVACÍ SUBSYSTÉMY II.

SBĚRNICOVÁ ORGANIZACE

- všechny jednotky jsou propojeny jednou nebo několika společnými sběrnicemi
- komunikace je rozdělena do diskrétních transakcí mezi vysílačem a přijímačem. Transakce začne, když některý účastník získá kontrolu nad sběrnicemi a stane se řídící jednotkou (master). Často existuje více jednotek, které se mohou stát řídící.

- ARBITRÁŽNÍ PROTOKOL

– je uplatnění, pokud více jednotek ve stejný okamžik usiluje o získání řízení sběrnice, transakce na sběrnici je řízena komunikačním protokolem

- ASYNCHRONNÍ PROTOKOL

– protokol, při kterém přednost elementu informace může začít kdykoli

- SYNCHRONNÍ PŘENOS

– může být zahájen jen v přesně stanovený okamžik. Provoz na sběrnici je vždy řízen taktovacím kmitočtem (hodinové impulzy). Výkon sběrnicového systému je určen dvěma parametry:

1, přenosovým časem
2, šířkou pásma (bity za sekundu)

ARBITRÁŽ POŽADAVKŮ O SBĚRNICI

- sběrnice je sdílené médium – dochází k situacím, kdy o použití soupeří několik jednotek. Pravidla při rozhodování o přidělování sběrnice jsou v zásadě dvojí:

1, centrální arbitráž – priorita, pořadí příchodu (LIFO) nebo náhodné

2, decentralizovaná arbitráž – prioritní linka (daisy chain), cyklické (tokem) a kolizní

- centrální arbitráž zajišťuje zvláštní jednotka, která vyhodnocuje přicházející žádosti

- decentralizovaná arbitráž používá rozhodovací algoritmy vestavěné do každé připojené jednotky. Jednotky mají vyšší inteligenci a potřebují jistý čas na provedení arbitráže a výměnu zprávy o jejím výsledku.

PROPOJOVACÍ SUBSYSTÉMY III.

- libovolný propojovací systém bude vyžadovat jistou programovou obsluhu. V případě srovnatelných rychlostí obou komunikujících jednotek bude pro programátora transparentní. Je – li rozdíl významný, je třeba komunikaci řešit tak, aby rychlejší jednotka byla komunikačním procesem zatížena co nejméně.

- existují následující způsoby práce:

- programová obsluha
- obsluha s využitím přerušení
- pomocí blokové přímého přístupu do paměti (DMA)
- samotné kanálové procesory

- mechanizmus přerušení je v zásadě vždy stejný – přerušení práce procesoru na vnější podnět, vykonání obsluhy přerušení a návrat k původní práci

- obvyklý způsob vyvolání programu obsluhy přerušení je nucený skok na adresu, kde je program uložen, pomocí tabulky vektorů, ukazující na příslušné obsluhy přerušení (skutečnost je komplikovanější)

- přenos se v počítači uskutečňuje buď po jednotlivých datových elementech (bajty) nebo po blocích. Pomalá zařízení (tj. taková, u kterých je odezva na přenos podstatně delší než vlastní přenos), používají většinou přenos po datových elementech (asynchronně) a rychlá pak blokový (synchronní) přenos

- BLOKOVÉ PŘENOSY – DIRECT MEMORY ACCESS (DMA)

- někdy je výhodné, aby se procesor řízení komunikačního procesu neúčastnil
  - procesor je řadičem DMA odpojen od sběrnice (a případně i zastaven)
  - řadič DMA zpomalí procesor zastavením hodin (kradení cyklů)

- KANÁLOVÉ PŘENOSY

- snaha o nezávislost komunikace na procesoru vedla ke vzniku komunikačních procesorů. Procesor je rychlým spojem propojen se specializovaným procesorem, kterému pouze předá požadavek na přemístění určitého množství dat

- komunikační procesory jsou speciálně navržená zařízení z vlastní instrukční sadou orientovanou na efektivní přesuny dat. Rychlému spoji říkáme komunikační kanál a komunikačnímu procesoru kanálový

PŘEHLED INSTRUKCÍ ASEMBLERU.

Sk	Kód	Operandy	Popis činnosti
I.	ld	ra, K	Naplnit registr z paměti, adresa je K
	ld	ra, [rb]K	Naplnit registr z paměti, adresa je K + obsah rb
	ldr	ra, D	Naplnit registr z paměti relativně (PC + D)
	st	ra, K	Uložit obsah registru do paměti, adresa je K
	st	ra, [rb]K	Uložit obsah registru do paměti, adresa je K + obsah rb
	str	ra, D	Uložit obsah registru do paměti relativně (PC + D)
	la	ra, K	Naplnit registr adresou
	la	ra, [rb]K	Naplnit registr adresou
	lar	ra, D	Naplnit registr adresou relativní (PC + D)
II.	add	ra, rb, rc	Sečíst rb + rc a uložit výsledek do ra
	addi	ra, rb, K	Přičíst konstantu k rb a výsledek uložit do ra
	sub	ra, rb, rc	Odečíst rc – rb a výsledek uložit do ra
	neg	ra, rc	Druhý dvojkový doplněk rc uložit do ra
	or	ra, rb, rc	Logický součet rb a rc, výsledek do ra
	ori	ra, rb, K	Logický součet rb a konstanty, výsledek do ra
	and	ra, rb, rc	Logický součin rb a rc, výsledek do rc
	andi	ra, rb, K	Logický součin rb a K, výsledek do ra
	not	ra, rc	Negace rc a uložit výsledek do ra
	shr	ra, rb, P	Posun rb doprava P x, výsledek do ra
	shr	ra, rb, rc	Posun rb doprava rc x, výsledek do ra
	shra	ra, rb, P	Posun aritmetický rb doprava P x, výsledek do ra
	shra	ra, rb, rc	Posun aritmetický rb doprava rc x, výsledek do ra
	shl	ra, rb, P	Posun rb doleva P x, výsledek do ra
	shl	ra, rb, rc	Posun rb doleva rc x, výsledek do ra
	shc	ra, rb, P	Rotace rb doleva P x, výsledek do ra
	shc	ra, rb, rc	Rotace rb doleva rc x, výsledek do ra
III	br	rb, rc, P	Skok na adresu v rb když rc vyhovuje P
	brl	ra, rb, rc, P	Uložení PC do ra a přechod na rb, když rc vyhovuje P
	br	rb	Přechod na adresu v rb
	brl	ra, rb	Uložit PC do ra a přechod na adresu v rb
	brlnv	ra	Uložit PC do ra
	brzr	rb, rc	Přechod na adresu v rb když v rc je nula
	brlzr	ra, rb, rc	Ulož PC do ra a přechod na rb když v rc je nula
	brnz	rb, rc	Přechod na adresu v rb když v rc není nula
	brlnz	ra, rb, rc	Ulož PC do ra a přechod na rb když v rc není nula
	brpl	rb, rc	Přechod na adresu v rb když rc > nula
	brlpl	ra, rb, rc	Ulož PC do ra a přechod na rb když rc > nula
	brmi	rb, rc	Přechod na adresu v rb když rc < nula
	brlmi	ra, rb, rc	Ulož PC do ra a přechod na rb když rc < nula

PAMĚTI I.

- paměť je zařízení, které je schopné přijmout informace, uchovat je po danou dobu a na požádání je vydat

- paměť je složena z adresovatelných jednotek – paměťových míst (buněk, lokací)

- paměť můžeme charakterizovat:

A, funkcí (čtení, zápis)
B, kapacitou (bity, bajty)
C, vybavovací doba, cyklus

ROZDĚLENÍ PAMĚTÍ PODLE PŘÍSTUPU K DATŮM:

RAM (random access memory) – paměť s libovolným přístupem. Doba odezvy je konstantní a nezávisí na adrese, která je použitá pro čtení a zápis

SAM (serial access memory) – paměť se sekvenčním přístupem – zpožďovací linky

DAM (direct access memory) – paměť s přímým přístupem (disk)

CAM (content addressable memory) – paměť adresovaná obsahem

ROZDĚLENÍ PAMĚTÍ PODLE IMPLEMENTACE:

RWM (read/write memory) – paměti pro čtení a zápis na libovolnou adresu

ROM (read only memory) – paměti schopné pouze číst

SRAM (static random access memory) – statické paměti RAM

DRAM (dynamic random access memory) – dynamické paměti RAM

PROM (programmable read only memory) – jendou programovatelné

PAMĚTI II.

HIERARCHICKÉ USPOŘÁDÁNÍ PAMĚTI:

REGISTRY – vysoká rychlost přístupu – realizace přímo jakou součást procesoru

CACHE – vyrovnávací paměť – co nejblíže procesoru (10 nanosekund)

HLAVNÍ – fyzická paměť počítače pro instrukce a data (10 – 100 nanosekund)

ODKLÁDACÍ – ukládání stránek hlavní paměti – disk (milisekundy)

SEKUNDÁRNÍ – vnější paměti s náhodným přístupem (disky, diskety)

ARCHIVNÍ – trvalé ukládání velkých objemů dat, ke kterým se přistupuje zřídka

ORGANIZACE PAMĚŤOVÝCH JEDNOTEK:

- paměti se širokým slovem – do datového registru se předává několikanásobek potřebné informace. Nutným předpokladem je následné čtení dat či instrukcí

- paměti s prokládanými cykly – rozdělení pamětí na moduly (banky) a střídavý přístup k informacím v různých modulech

KONTROLA A KOREKCE CHYB

- parita příčná a podélná. Samoopravné kódy (Hammingův)

PAMĚTI III.

HIERARCHICKÉ USPOŘÁDÁNÍ PAMĚTI

- vychází ze skutečnosti, že přístupy do paměti mají tendenci shlukovat se do skupin

- lokalita referencí:

A, časová – reference, která byla právě použita a bude v krátké době použita znovu

B, prostorová – reference následující bude pravděpodobně ležet poblíž právě použité

- mechanizmus funkce je následující – paměť je rozdělena na úseky, tvořené posloupnosti adresovatelných jednotek. Podle požadavků procesoru je obsah hierarchicky nižší paměti vyměněn s vybranou oblastí hierarchicky vyšší paměti

- mapovací mechanizmy – stránkovaná a segmentovaná paměť

NĚKTERÉ DŮLEŽITÉ ORGANIZACE VÝBĚRU Z PAMĚTI

- zásobníková organizace (STACK) je pamětí LIFO. Může být realizovaná jako zvláštní součást operační paměti i jako samostatná paměť na bázi registrů. Je adresovaná přes zvláštní registr – SP (stack pointer). Základními operacemi se zásobníkem je operace uložení a vybrání položky (PUSH, POP)

OCHRANA PAMĚTI

- mechanizmus řízení přístupu ke specifickým částem operační paměti. Narušení ochrany se projeví jako přerušení, které obsluhuje operační systém. Užívá se v jednouživatelském i víceuživatelském režimu práce.

- může být realizováno takto:

- mezními registry – obsah MAR se porovnává s limitními hodnotami
- maskou – paměť je rozdělena na stránky. Maska = číslo stránky
- klíče – prakticky totéž co maska. Srovnání provádí software
- metabity – pomocné bity přidané k datovému obsahu paměti.

PC – PAMĚTI

- systémová paměť může být tvořena dvěma jakostními typy fyzické paměti –

- DRAM (DYNAMIC RAM)
- SRAM (STATIC RAM)

- paměťové buňky čipu DRAM jsou tvořena dvojicí malého kondenzátoru a tranzistoru
- paměťové buňky čipu SRAM jsou tvořeny 6 tranzistory SRAM = 2M = DRAM 64 MB

ZÁKLADNÍ TYPY PAMĚŤOVÝCH MODULŮ DRAM

- FPM – Fast page mode

- EDO – Extended data ant

- SDRAM – Synchronous dynamic RAM

- RDRAM – Rambus DRAM

- DDR SDRAM – Double data rate SDRAM

FYZICKÉ USPOŘÁDÁNÍ PAMĚTI RAM

- SIMM – single inline memory module – 30 a 72 vývodů

- DIMM – dual inline memory module – 168 a 1284 vývodů (DDRDRAM)

- RIMM – Rambus inline memory module – 184 vývodů

BUŇKY PAMĚTÍ

- nejmenší množství paměti, které může být najednou adresováno

- šířka paměti musí vždy odpovídat šířce procesorové sběrnice

VÝVODY MODULŮ

- povrch kontaktů je pokryt zlatem nebo cínem. Nelze doporučit používání konektorů a vývodů modulů s různým pokovením (fretting corrosion)

SPOLEHLIVOST PAMĚŤOVÉHO SUBSYSTÉMU

- chyby můžeme rozdělit na logické a fyzické

- FYZICKÉ – projeví se trvalým selháním modulu

- LOGICKÉ – projeví se nepravidelně a mohou být způsobeny : poruchami napájení, rušením,
statickými výboji, zářením nebo nevhodným použitím paměťového modulu
(výpadky časování)

- ochranou je použití paritního bitu nebo ECC (Error correcting code), což zvyšuje cenu paměťového modulu

PC – SYSTÉMOVÉ SBĚRNICE

- sběrnice je definována, jako skupina vodičů, po které jsou data přenášena mezi subsystémy základní desky. Většinou systém obsahuje několik sběrnic, které jsou uspořádány hierarchicky. Pomalejší sběrnice je vždy zapojena do rychlejší, která je v hierarchii o úroveň výše. Obvody čipové sady pak vykonávají funkci mostu mezi sběrnicemi.

ZÁKLADNÍ SBĚRNICE PERSONÁLNÍHO POČÍTAČE

- PROCESOROVÁ SBĚRNICE

- propojení procesoru s paměti cache a systémovou pamětí. Nazývá se také FSB (front side
bus). Pracuje na rychlosti 66,100,133,200 Mhz a má šířku 64 bitů (8 bajtů). S ostatními částmi základní desky je propojena severním kostem nebo rozbočovačem paměti. Je to nejrychlejší
a hierarchicky nejvyšší sběrnice

- GRAFICKÝ PORT

- propojení severního mostu a grafického subsystému. Není sběrnicí v pravém slova smyslu.
AGP (accelerated graphic port). Pracuje na rychlosti 66 Mhz a má šířku 32 bitů (4 bajty). V
závislosti n verzi může být v jednom cyklu převeden 1,2 nebo 4 bajty. Efektivní přenosová
rychlost je až 1066 MB/s. V zásadě platí 2 funkce : propojuje a také umožňuje grafické kartě

přístup do systémové paměti. (redukuje nároky na samostatnou video paměť)

- SBĚRNICE PCI

- zabezpečuje zapojení rozšiřujících desek do systému (peripheral component interconnect)
Pracuje na rychlosti 33 Mhz a má šířku 32 bitů. Existuje i rychlejší varianta – 66 Mhz a šířka
64 bitů. Řadičem sběrnice je severní most. Sběrnice se také nazývá mezanínová. Její
zavedení změnilo koncepci PC.

- SBĚRNICE ISA

- nejpomalejší sběrnice, která slouží k zapojení rozšiřujících desek do systému. (industry
standart architekture). Pracuje na frekvenci 8 Mhz a má šířku 16 bitů. Reálná přenosová
rychlost bývá poloviční. Řadičem sběrnice je jižní most

- existují i jiné sběrnice: MCA, EISA, VESA (lokální sběrnice), PC – card (dříve PCMCIA), FireWire (IEEE – 1394), USB – univerzální

- na základní desce mohou být i jiné konektory AMR – audio modem riser, CNR – communicatiion and networking riser – toto nejsou žádná zakončení paměti

ZOBRAZENÍ DAT V POČÍTAČI I.

- počítač je stroj, který je schopen přijmout informace, aplikovat... Informace je v počítači uložena v paměti, přemísťována pomocí komunikačních kanálů a přeměněna v procesoru

- stroj musí pracovat s nějakou symbolikou, která bude:
- srozumitelná pro stroj
- reprezentovaná pomocí fyzikálních veličin
- převoditelná do symbolů srozumitelných člověku
- dostatečně obecná i pro popis složitých nehmotných představ

- data jsou v počítači zapsána pomocí kódu. Z hlediska organizace počítače dělíme kódy na:

- komunikační (zde se sleduje především odolnosti vůči chybě při přenosu)

- kódy pro provádění matematických operací (sledujeme především jednoduchost provádění úkonů, účinnost zobrazení)

- základním objektem, se kterým je v počítači manipulováno je ***datový typ.***

- místo uložení

- rozsah možných hodnot (množina povolených hodnot)
- soubor pravidel pro transformace (množina povolených transakcí)
- soubor pravidel definující způsoby a oprávnění přístupu

- každý datový typ musí explicitně (přímo vyjádřeno) nebo implicitně (vyplývá) obsahovat výše uvedené vlastnosti

ZÁKLADNÍ DATOVÉ TYPY:

- logická hodnota (0 = false, cokoli = true)
- znak abecedy
- číslo

- literál – je lexikální jednotka, která přímo reprezentuje hodnotu
- je hodnota definována sama sebou (konstanta)

ZOBRAZENÍ DAT V POČÍTAČI II.

- informace je v počítači reprezentována množinou bitů a kódována

- kód přenese příslušnou informaci na data v binární soustavě vhodnou k zpracování strojem

- k reprezentaci alfanumerických symbolů se používají kódy zadané tabulkou

EBCDIC – extended binary coded decima interchange code
ASCII – american standart code for information interchange
UNICODE – kód, který se snaží zavést konsurcium firem

- základem je čtyřbitový váhový kód BCD pro desítkové číslice
- rozšířením na 8 bitů = 256 možností. ASCII prakticky totéž, ale původně jen 7 bitů = 128
možností
- rozšířením na 8 bitů dalo možnost přidat národní abecedy (Latin 2)

ČÍSELNÍ SOUSTAVA

- rozumíme soubor určitých znaků a pravidel sloužících k zobrazení čísla. V zásadě známe:

A, poziční (polyadické) – význam znaku závisí na místě
B, nepoziční (nepolyadické) – znak je na pozici nezávislý

- nejznámější je polyadická soustava dekadická nebo binární. Hexadecimální nebo oktalyová je jen zkrácený zápis binární soustavy

- pro zobrazení čísel v počítači používáme polyadickou binárně kódovanou soustavu

- číslo může být reprezentováno:
- přirozeným tvarem – pevná řadová čárka – počet řádků je konstantní
- semilogaritmickým tvarem – pohyblivá řadová čárka – počet řádů mantisy a charakteristiky je pevně stanoven (čísla REAL)

- nezávisle na kódu je základní vlastností zobrazení čísel konečný počet zobrazitelných míst
(256, 65536, 4.294.367.296)

- volba reprezentace dat je jedna z nejdůležitějších etap návrhu architektury počítače, protože zásadním způsobem ovlivňuje i organizaci stroje.

ZOBRAZENÍ DAT V POČÍTAČI III.

- pro zobrazení čísel v pevné řadové čárce jsou většinou používány jednoduché váhové binární kódy. Číslo je reprezentováno kombinací bitů, kde je každému bitu přidělena jeho váha.

- řadová čárka bývá nejčastěji úplně vpravo – obecně může být umístěna kdekoli – závisí to na architektuře počítače. Výhodou této reprezentace je jednoduchost konstrukce a shodnost realizace pomocí elektronických obvodů

BINÁRNÍ ČÍSLA SE ZNAMÉNKEM

(existuje několik způsobů reprezentace)

- přidání znaménkového bitu
- zobrazení s posunem – přičtení pozitivní konstanty (AD převodníky)
- dvojkový doplněk – negace + 1

BCD čísla

(binary coded decimal)

- přímý binární kód, kde jsou dekadické číslice ukládány do čtveřic bitů, které reprezentují
právě jednu dekadickou číslice

- VÝHODY: nejsou nutné převody – odstranění nepřesnosti a konverzních chyb

- NEVÝHODY: asi 20% renundance. Znaménko: +1100, -1101 (jsou volné kombinace)

- chybové stavy – přetečení a podtečení

- při práci v pevné řadové čárce je počet bitů ve slovně pevně stanoven a tím omezuje počet zobrazitelných čísel (aritmetiku čísel s konečnou délkou)

- je potřeba sledovat nejen velikost operandů, ale i pořadí prováděných operací.

- neplatí obecně zákon asociativní a distributivní
PŘ.: 700+(400-300) = (700+400) – 300 !!!!!! na 3 místa

ZOBRAZENÍ DAT V POČÍTAČI III.

- čísla v plovoucí řádové čárce mají tyto vlastnosti:

- volný rozsah zobrazitelných čísel
- čísla v semilogaritmickém tvaru netvoří kontinuum (existuje mezera)
- stejná přesnost pro všechna čísla
- nerovnoměrné pokrytí číselné osy
- nutnost zaokrouhlení nevyjádřitelných čísel

- existuje několik způsobů zobrazení čísel s plovoucí řádovou čárkou:

- přímé – mantisa i exponent jsou opatřeny znaménky uloženými zcela vlevo, řádové čárky
jsou zcela vpravo

- nejsrozumitelnější zobrazení, nejpřesnější pro celá čísla

- exponent s posuvem

- se skrytým bitem – musí být vždy normalizované

- správná volba mantisy a poměru mantisy k exponentu má klíčový význam při návrhu architektury stroje. Pro standardizaci práce s čísly v plovoucí řádové čárce bylo proto vytvořeno doporučení

IEEE-754. Není závazné, ale většina výrobců jej dodržuje

PROCESORY INTEL

- jednoduchá přesnost : 32 bitů 0,1.2x10-38 3.4x10+38
- dvojitá přesnost : 64 bitů 0,2.3x10-38 1.1x10+308
- zvětšená přesnost : 80 bitů 0,3.4x10-4932 1.1x10+4932
  - posunutí exponentu 7F, 3FF, 3FFF

- existují i hodnoty + - nekonečno a NaN (Not a Number) a nenormalizované (tiny) čísla

DATA S AUTOMATICKOU IDENTIFIKACÍ

- tagované paměti – snadná korekce chyb, konverze, obecné instrukce
- deskriptory dat – obsahují informace o typu, přesnosti, rozměrech, umístění

ZOBRAZENÍ DAT V POČÍTAČI – REÁLNÁ ČÍSLA

ČÍSLO = (-1) ^ sign x 2 ^ (exponent – 127) x (1.mantisa)

4 0 A 0 0 0 0 0

0 | 100 0000 1 | 010 0000

1 x 4 x 1.25 = 5

4 2 F E

0 | 100 0010 1 | 111 1110

1 x 64 x 1.98 = 63

4 2 7

0 | 100 0010 0 | 111

1 x 32 x 1.875 = 60

4 1 2 8

0 | 100 0001 0 | 010 1000

1 x 8 x (1/4 + 1/16) = 1 x 8 x 1.3125 = 10.5

ZOBRAZENÍ DAT V POČÍTAČI IV.

Binární čísla bez znaménka

bajt	8 bitů	byte
slovo	16 bitů	word
dvojslovo	32 bitů	doubleword (486)
čtyřslovo	64 bitů	quadword
dvojité čtyřslovo	128 bitů	double quadword (PIII)

Celá čísla kladná nebo záporná

položka	bez znaku	-	+
bajt	255	128	127
slovo	65.536	32.768	32.767
dvojslovo	4.294.967.295	2.147.483.648	2.147.483.647
čtyřslovo	1.844x10^19		

Reálná čísla

položka	bitů	znak	j-bit	mantisa	exponent	řádů
jednoduchá přesnost	32	1	Ne	24	8	127
dvojitá přesnost	64	1	Ne	52	11	1.023
rozšířená přesnost		80	1	Ano 63	15	16.383

- vícebajtové datové položky jsou v paměti zapsány ve formátu méně významný bajt na nižší adrese (little endian). Platí pouze pro architekturu 1A-32 (neplatí paušálně u Itania). Datové položky mají být v paměti počítače vyrovnány na přirozenou hranici typu. Tím se rozumí, že slovo by mělo ležet na adrese dělitelné dvěma, dvojslovo na adrese dělitelné čtyřma a čtyřslovo na adrese dělitelné osmi. Nerespektování této skutečnosti v lepším případě způsobí zmenšení výkonu stroje – procesor bude potřebovat dva přístupy do paměti k přičtení takto uložené datové položky. V horším případě (záleží na instrukci) to může způsobit chybový stav (#GP –general – protection Exception). Tento fakt je důležitý pro programátora v asembleru, ve vyšším programovacím jazyku je to věcí překladače.

BINÁRNÍ ARITMETIKA – SHRNUTÍ

- při sčítání a odčítání binárních čísel platí následující

0+0=0	0-0=0
0+1=1	0-1=1 a výpůjčka
1+0=1	1-0=1
1+1=0 a přenos	1-1=0

```
 01101        110110
 10111      -  10111
100100       1010111
```

ZÁPORNÁ ČÍSLA :

znaménkový bit – nevýhodné pro stroj
prvý dvojkový doplněk – nevýhodné pro člověka, jsou dvě nuly
druhý dvojkový doplněk – nevýhodné pro člověka, asymetrický rozsah

Při manipulaci s binárními čísly ve dvojkovém doplňku platí následující pravidla:
1, mají – li čísla různá znaménka, potom k přeplnění nemůže dojít
2, dojde – li k přeplnění, potom výsledek má opačné znaménko jako operandy

Reálná čísla ve formátu s pohyblivou řádovou tečkou – standard IEEE-754. Tento standard specifikuje následující:

- způsob zobrazení reálných čísel v počítači
- přesnost výsledků aritmetických operací a operací srovnání
- konverze mezi celými a desetinnými čísly
- konverze mezi reálnými čísly s různou přesností
- definuje podmínky vzniku zvláštních situací a způsob manipulace s nimi

ZVLÁŠTNÍ HODNOTY:

Nula
– není přímo zobrazitelná (skrytý bit je vždy 1). Je definována jako exponent i zlomková část samé nuly. Existuje kladná a záporná nula – obě mají stejnou platnost

Nekonečno
– je reprezentováno jako exponent samé 1 a zlomková část samé nuly. Existuje kladné i záporné nekonečno. Norma definuje výsledky operací s nekonečnem a nulou

Nečíslo
– NaN je speciální hodnota, která vznikne jako výsledek operací matematicky nedefinovaných (0/0,nekonečno-nekonečno,nekonečno x 0, nekonečno/nekonečno)

PROCESOR INTEL 4004

- prvý procesor Intel
- vyroben technologií PMOS
- tloušťka čáry 10 μm
- obvod tvořen 2300 tranzistory na křemíkové destičce 3x4mm
- frekvence 108 kHz
- výpočetní výkon 60000 instrukcí za vteřinu
- instrukční soubor – 46 instrukcí
- napájecí napětí 15 V, příkon 1 W
- pouzdro DIP 16
- vyroben na zakázku japonské firmy BUSICOM
- sběrnice je 4bitová, obousměrná, přepínavá (data/adresy)
- dělená paměť pro data a instrukce
- 16 4bitových nebo 8 8bitových registrů
- akumulátor je 4 bitový
- příznakové bity přenosu a testování provedeny jako separátní bity akumulátoru
- programový počítač – 12 bitů
- Instrukční cyklus 10.8 μs
- operace součtu dvou 8 bitových čísel trvala 850 μs
- vykonání 4-bitové instrukce na 8 taktů hodin : 3 + 2 +3

- čip CPU je základem obvodové sady MCS – 4 (Microcomputer System 4 – bit). Sadu tvoří obvody 4001-ROM 256x4, 4002-RAM 4x20x4, 4003 – posuvný registr 10 bitů (slouží pro vstup/výstup). Minimální konfigurace počítač : CPU a jeden obvod paměti ROM

mikroPROCESORY INTEL 8086 a 8088

ZÁKLADNÍ VLASTNOSTI:

- úplná 16bitová paralelní univerzální procesorová jednotka
- vyroben unipolární technologií HMOS
- pouzdro DIL 40
- ekvivalent cca 29000 tranzistorů
- napájení +5 V, 275 mA max
- vstupy a výstupy TTL úrovně
- hodinový signál jednofázový

- 20bitová adresová sběrnice dovoluje adresovat 1M paměti
- 16bitová interní a 16bitová (8bitová u 8088) externí datová sběrnice
- základem architektury jsou 2 zřetězené subprocesory. Pracují poměrně nezávisle a využívají překrytí fáze zápisu, čtení i výběru instrukce s fázemi vykonání instrukce předchozí. Výsledkem je zvětšení rychlosti a zmenšení nároků na rychlost hl. paměti

dvě skupiny zásobníkové paměti:

- registry pro všeobecné použití (skupina HL)

- 4 registry, každý adresovatelný jako jeden 16bitový nebo dva nezávislé 8bitové

- skupina ukazatelů a indexových registrů (skupina PI)

- mohou se podílet na většině aritmetických a logických operací. Všechny registry vyhovují pojmu střadač

- výkonná jednotka má 9 jednobitových indikátorů uspořádaných v 16bitovém slově

15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
				OF	DF	IF	TF	SF	ZF		AF		PF		CF

CF – přenos (carry)
PF – parita výsledků je sudá (parity)
AF – přenos z nižšího nibble AL registru (auxiliary)
ZF – výsledek je nulový (zero)
SF – výsledek je záporné znaménko (sign)
OF – aritmetické přetečení (overflow)
DF – autodekrementace (direction)
IF – povolení přerušení (interrupt)
TF – krokovací režim (trap)

SEGMENTACE PAMĚTI

PŘÍKLADY :

CS	1	0	0	0	
IP		2	2	1	3
	1	2	2	1	3

DS	2	0	0	0	
SI		2	2	1	3
	2	2	2	1	3

SS	3	0	0	0	
SP		F	F	F	F
	3	F	F	F	F

- paměť je bajtově orientovaná – základní adresovatelnou jednotkou je jeden bajt. Slovo o délce 16 bitů lze ukládat na libovolné adresy. Uloží se tak, že nižší bajt přijde na specifikovanou adresu a vyšší bajt na následující vyšší adresu. Doporučuje se však slova přednostně ukládat na sudé adresy – čtení i zápis slova se realizuje v jednom cyklu paměti (při liché adrese je na tutéž operaci třeba dvou cyklů paměti)

PROCESORY INTEL 8086

SHRNUTÍ:

- procesor představuje 16bitový mikroprocesor (vnitřní sběrnice je 16bitová)
- vnější datová sběrnice je 16bitová
- adresová sběrnice je 20bitová (adresuje 1M)
- vnitřně je realizován jako dvě nezávislé jednotky
- cílem je zvýšení výkonu procesoru

JEDNOTKA BIU

- zajišťuje styk s vnější sběrnicí, výpočet adresy a asynchronní výběr instrukcí (realizuje instrukční fázi VYTÁHNI)

- funkčními bloky jednotky jsou:

- 20bitová sčítačka pro výpočet adresy (offset + segment) = adresa
- programový čítač
- vnitřní komunikační registry
- čtyři segmentové registry (CS, DS, ES, SS)
- logika řízení vnější sběrnice a fronta připravených instrukcí (realizována jako 6-ti bajtová paměť FIFO)

VÝKONNÁ JEDNOTKA EU

- vykonává připravené instrukce

- funkčními bloky jednotky jsou:

- ALU s třemi registry pro dočasné uložení operandů

- registr příznaku

- skupina registrů pro všeobecné použití (čtyři registry HL ,dva registry ukazatelů,dva indexové registry), logika instrukcí provedená jako mikroprogram, logika řízení vnitřní sběrnice a vnitřní synchronizace, který mění vnější hodinový signál v posloupnost interních řídících impulsů

- instrukční soubor obsahuje 133 instrukcí, které dovolují zpracovávat jak 8bitové tak i 16bitové datové položky. Instrukce může mít jeden, dva nebo žádný operand. Délka instrukce je proměnná v rozmezí od 1 do 6 bajtů. Prvý bajt vždy obsahuje instrukční kód, ostatní bajty nesou informace o adresách a operandů. Prvý bajt také obsahuje speciální pole (bit W, = 1 slovo) určující, zda instrukce zpracovává bajt nebo slovo. Procesor je schopen realizovat většinu známých adresový módů. Použití segmentových registrů je předdefinováno. Speciálním prefixem lze předepsat (poručit) jiný způsob (například adresu datové položky nikoli k registru DS, ale třeba CS). Způsob adresace operandů určuje (většinou) druhý bajt

instrukce. Obsahuje pole režimu (mód, 2 bity), což určuje, je-li použito posunutí. Pole registr
obsahuje číslo registru (malou adresu, 3bity). V poli r/m je zakódováno, jakým způsobem se
vypočte efektivní adresa. Je očividné, že doba vykonání jedné a téže instrukce, závisí na
způsobu výpočtu efektivní adresy.

Počet taktů potřebných při různých způsobech adresace:

- přímá 6
- bázová 5
- báze + posunutí 1
- báze + index 7
- báze + index + posunutí 11

PROCESOR INTEL 8086

PŘERUŠOVACÍ SYSTÉM

- přerušení je obecně asynchronní událost, která způsobí pozastavení probíhajícího procesu a umožní spuštění procesu jiného (proces obsluhy přerušení)

- zdroj signálu inicializujícího přerušení může být:

vnější (třeba z periferie)
vnitřní (softwarové přerušení)

- po příchodu signálu přerušení je běžná sekvence instrukcí dočasně přerušena řízení je tím předáno rutině obsluhy (interrupt handler).

- obsluha přerušení vyžaduje provedení určitých operací navíc (režim přerušení), což spotřebovává jak paměť, tak čas. Nákladné je hlavně zabezpečení neměnnosti pracovního prostředí přerušeného procesu (uložení kontextu), tak, aby byla zajištěna kontinuita procesu (protože procesor 1-86 má jen jednu sadu registrů, je třeba jejich obsah uložit a před návratem k původnímu procesu opět obnovit).

- určitou dobu také trvá samotná operace přepnutí procesů (pro 1-86 od 50 až 61 hodinových cyklů). Zdrojů přerušení je většinou vícero. Každý zdroj přerušení má svou obslužnou rutinu. Adresy vstupních bodů těchto rutin se nazývají vektory přerušení a jsou uspořádány do tabulky uložení v paměti od adresy nula. Vnější přerušení z různých zdrojů mohou běžně vzniknout ve stejný časový okamžik. Pořadí důležitosti, v jakém se přerušení obsluhuje určuje priorita přerušení a je řízena hardwarově (PIC).

HARDWAROVÉ PROVEDENÍ

- procesor 1-86 má pro příjem signálů vnějšího přerušení dva vývody.

Vývod 17 (NMI - nemaskovatelné přerušení)

- slouží k signalizaci fatálních událostí. Signál nelze ignorovat (zamaskovat, odstínit), protože oznamuje katastrofické stavy systému (chyba parity paměti, sběrnice), na které procesor musí okamžitě reagovat (obvykle chybová hláška a HALT)

Vývod 18 (INTR)

- na tento vývod je připojen programovatelný kontrolér přerušení (PIC). Má 8 vstupů a obvody lze je řadit do kaskády (běžně 2, maximálně až 64). Další přerušení realizované hardwarově slouží k ladění. Nastavením TF v registru příznaků přijde procesor do krokovacího režimu což znamená, že po každém vykonání instrukce dojde k přerušení. Obslužná rutina aktivuje ladící systém, který stanoví, co se vykonáním instrukce

změnilo (a oznámí to uživateli). Další čistě hardwarové přerušení je reakce na dělení
nulou a přetečení.

- tabulka rezervovaných vektorů přerušení (firma Intel) obsahuje 32 vektorů (32 rutin obsluhy související se stavem procesoru, adresy 0 až 7 FH). Nejdůležitější jsou následující: 0 – dělení nulou, 1 – krokování (trap), 2 – NMI, 3 – ladění (breakpoint), 4 – přetečení (overflow)

PŘERUŠOVACÍ SYSTÉM – POPIS FUNKCE

- při vzniku přerušení se provedou následující operace :

- dokončí se rozpracovaná instrukce, přerušení se uplatní až po skončení instrukce. Uloží
se do zásobníku registr příznaků.

- v registru příznaků se vynulují bity IF a TF.

- do zásobníku se uloží registr CS, který se naplní z adresy N*4+2

- do zásobníku se uloží registr IP, který se naplní z adresy N*4

- programátor v rutině obsluhy přerušení musí uchovat obsah všech registrů, které hodlá
použít (modifikovat obsah). Přerušovací systém je možno při obsluze přerušení aktivovat
(instrukce STI) a dovolit přerušení obsluhy přerušení. Návrat k přerušenému procesu se
provede instrukcí IRET. Ta obnoví ze zásobníku IP, CS a registr příznaků (to uvede přerušovací systém do původního stavu)

KÓD	***TYP***	***FUNKCE***
0	P	Dělení nulou
1	P	Ladění – krokování
2	P	Nemaskovatelné přerušení NMI
3	P	Ladění – breakpoint
4	P	Aritmetické přetečení
5	S	Print screen
6	P	Chybný operační kód
7	P	Koprocesor nepřítomen
8	H	IRQ 0 – časovač
9	H	IRQ 1 – klávesnice
A	H	IRQ 2 – kaskáda
B	H	IRQ 3 – sériový adaptér 2 nebo 4
C	H	IRQ 4 – sériový adaptér 1 nebo 3
D	H	IRQ 5 – tiskárna LPT2
E	H	IRQ 6 – kontrolér floppy disku
F	H	IRQ 7 – tiskárna LPT1

10	S	Služby video
11	S	Seznam zařízení
12	S	Velikost operační paměti

REZERVOVANÉ OBLASTI PAMĚTI:

- některé adresové lokace paměti jsou rezervované pro speciální operace CPU. V dolní části
paměti RAM adresy 0-3FFH je vyhrazená oblast pro vektory přerušení. Celkem 255 vektorů x 4 bajty = 1KB. Podobně v každé horní části od adresy FFFF=0H do konce paměti je prostor 16 bajtů pro RESET BOOTSTRAP. Klasické IBM PC má 640 KB paměti RAM. Zbytek do 1 MB je vyhrazen pro adaptery zařízení a moduly ROM (BIOS)

START MIKROPROCESORU:

- se vzestupnou hranou signálu RESET procesor ukončí svou činnost a provede:

  - vyčištění fronty instrukcí

  - vynulování registrů příznaků a DS, ES, SS a IP, do registru CS se zapíše hodnota FFFFH

  - sestupnou hranou signálu RESET mikroprocesor zahájí činnost vždy od adresy FFFFOH (CS = FFFFH a IP = 0000). Délka impulsu RESET je minimálně 4 hodinové cykly

VSTUPY A VÝSTUPY

- mikroprocesor pro styk s vnějšími zařízeními používá V/V brány adresové odděleně od hlavní
paměti (isolated IO). Může komunikovat buď po 8 nebo 16 bitech. Má přístup k 64K 8bitových
nebo 32K 16 bitových bran. Pro adresy bran (sudá-lichá) platí stejná doporučení a omezení
jako při adresování paměti.

ZPŮSOBY ADRESACE OPERANDŮ

1, Přímý (bezprostřední) operand – konstanta je součástí instrukce

```
mov  ax, 12h
```

2, Přímá adresa – hodnota offsetové části adresy je součástí instrukce

```
mov  ax, ADR1
mov  ax, ds:[100]
```

3, Nepřímá adresa – offsetová část adresy je uložena na adrese, která je součástí instrukce

```
mov  ax, [bx]
mov  ax, [ADR1]
```

4, Bázová adresa – získá se jaku součet přímé a nepřímé adresy v bp,bx

```
mov  ax, [bx]+4
mov  ax, [ADR1]+4
```

5, Indexová adresa – součet přímé adresy a obsahu indexového registru si,di

6, Kombinovaná adresa – báze + index

```
mov ax, [bx+di]
```

7, Kombinovaná adresa přímá + báze + index

```
mov ax, [bx+si] + 4
mov ax, [bx+si] + ADR1
```

- v případě lze pro danou instrukci zrušit implicitní přiřazení segmentového registru

```
mov ax, es: [bx+si] + 8
```

FYZICKÁ ADRESA – 20 bitů

LOGICKÁ ADRESA – CS + IP = 32 bitů

REGISTRY – 16 bitů

OFFSET : 0 – 65.535

SEGMENT 16 – 1M

mikroPROCESORY INTEL 8086 A 8088

INSTRUKCE

OPCODE	CÍL	ZDROJ

OPCODE – co se má udělat
OPERANDY – s čím se to má udělat

ZPŮSOB ADRESACE OPERANDŮ

- bezprostřední mov al, 12H bo 12
- registrová mov al, bl 80 d8
- přímá mov [550H], al a2, 5005
- registrová nepřímá mov dl, [si] 80a14
- bázovaná mov ax, [bx+4] 8b4704
- indexová mov [di+8], bl 885d08
- kombinovaná – báze + index mov [bp+si], ab 8822
- kombinovaná – báze + index + posunutí mov cl, [bx+di+2] 8a4902

ROZDĚLENÍ INSTRUKCÍ PODLE ÚČELU:

- instrukce pr manipulaci s daty
- aritmetické instrukce
- instrukce pro manipulaci s bity
- instrukce pro řízení chodu programu
- řídící instrukce pro procesor

INSTRUKCE PRO MANIPULACI S DATY:

- pro přesun dat
- pro práci s řetězci dat
- pro práci se zásobníkem
- pro obsluhu vstupu a výstupu

- s výjimkou instrukcí pro práci s řetězci a v/v je možno použít libovolný způsob operace

INSTRUKCE PŘESUNU DAT

PRO VŠEOBECNÉ POUŽITÍ:

- 1, instrukce přesunu dat — mov cíl zdroj
- 2, instrukce uložení do zásobníku — push zdroj
- 3, instrukce vyjmutí ze zásobníku — pop cíl
- 4, instrukce záměny — xchg cíl zdroj
- 5, instrukce transformace podle tabulky — xlat (al = as: [bx+al])

PRO PRÁCI S ADRESAMI:

- 6, uložení offsetové části adresy — lea cíl zdroj
- 7, naplnění adresou – segment ds — lds cíl zdroj
- 8, naplnění adresou – segment es — les cíl zdroj

PRO PRÁCI S REGISTREM PŘÍZNAKU:

- 9,
-10,
- 11,
- 12,

PRO OPERACE VSTUPU A VÝSTUPU:

- 13, vstup dat z portu — in cíl zdroj
- 14, výstup dat na port — out port zdroj

INSTRUKCE PRO PRÁCI S ŘETĚZCI ZNAKŮ

REP	repeat until		cx <>0
REPE	repeat while	equal and	cx <>0
REPZ	repeat		
REPNE	repeat while	not equal and	cx <>0
REPNZ	repent		
MOVS	copy	es : di < - ds : si	
MOVSB	no operands		
MOVSBV	no operands		
COMPS	compare	es : di < - ds : si	
COMPSB	no operands		
COMPSV	no operands		
SCAS	compare	es : di - ax	
SCASB	no operands		
SCASV	no operands		
LODS	copy	ax < - ds : si	
LODSB	no operands		
LODSV	no operands		
STOS	copy	es : di < - ax	
STOSB	no operands		
STOSV	no operands		

CLD, STD Set / reset direction flag (+,-si,di)

```
EXAM 1 :    cld
            mov cx, 16
            lea si, BLC1
            lea di, BLC2
       rep movs B

EXAM 2 :    cld
            xor ax, ax
            mov cx, 255
            lea di, BLC1
$1          stosB
            inc al
            loop $1
```

INSTRUKCE PRO PRÁCI S BITY

Logical	destination, source
AND	Finde the logical AND two operands 2-7
OR	Performs logical inclusive OR operation 2-7
XOR	Performs logical exclusive OR operation 2-3
NOT	Reverse each bit of byte, word 2-7
TEST	Performs logical compare operation 2-6
BSF	Bit scan forward 10+3n
BSR	Bit scan reverse 10+3n
BT,C,S,R	Bit test 3.13

Shift	destinatiion, count
SAL	Shift arithmetic left 2-5
SAR	Shift arithmetic right 2-5
SHL	Shift logical left 2-5
SHR	Shift logical right 2-5

Rotate	destination, count
ROL	rotate left 2-5
ROR	rotate right 2-5
RCL	rotate left through carry 2-5
RCR	rotate right through carry 2-5
SHLD	double precisiion shift left 3-7
SHRD	double precisiion shift right 3-7

EXAM 1 : shrd ax, bx, 12 ; double precision shift right

```
EXAM 2 :    mov cl,2
            shl ax, cl ; multiply an unsigned number by 4
            sal ax, cl ; multiply a signed number by 4
            shr ax, cl ; divide an unsigned number by 4
            sar ax, cl ; divide a signed number by 4

EXAM 3 : ; multiply by 10
            move bx, ax ; save contents of ax
            shl ax, 1 ; multiply by 2
            shl ax, 1 ; multiply by 4
            add ax, bx ; multiply by 5
            shl ax, 1 ; multiply by 10
```

INSTRUKCE ŘÍZENÍ CHODU PROGRAMU

ja	jnbe	>	if above	/ not below or equal
jae	jnb	> =	if above or aqual	/ not below
jb	jnae	<	if below	/ not above or equal
jba	jna	< =	if below or equal	/ not above
je			if carrz	
je	jz	= 0	if equal 0	
jg	jnle	> +	if greater	/ not less or equal
jge	jnl	> = +	if greater or equal	/ not less
jl	jnge	< +	if less	/ not greater or equal
jle	jng	< = +	if less or equal	/ not greater
jnc			if no carry	
jne	jnz	≠ 0	if not equal 0	
jno			if not overflow	
jnp	jpo		if not parity	/ parity add
jns		+	if not sign	
jo			if overflow	
jp	jpe		if paritz	/ paritz even
js			if sign	

INSTRUKCE CYKLU

jc x 2	= 0	if cx equal 0
loop	≠	loop while cx not equal 0
loope loopz	= 0	loop while cx equal 0
loopne loopnz	≠ 0	loop while not equal

INSTRUKCE SKOKU

jmp	jump unconditionally

INSTRUKCE ŘÍZENÍ CHODU PROGRAMU II

INSTRUKCE VOLÁNÍ PODPROGRAMU

call	call procedure
ret	return from procedure
retn	return from near procedure
retf	return from far procedure

INSTRUKCE ŘÍZENÍ PROCESORU

1, INSTRUKCE PRO PRÁCI S REGISTREM

cle	clear carry
cld	clear direction (auto – increment)
cli	clear interrupt - enable flag
cmc	complement carry
stc	set carry flag
std	set direction (auto – dekrement)
sti	set interrupt – enable flag

2, INSTRUKCE ŘÍZENÍ STROJE

hlt	halt procesor
lock	lock the bus
wait	wait to coprocesor
esc	escape to coprocesor
nop	no operation

INSTRUKCE PŘERUŠENÍ CHODU PROGRAMU

int	call to interrupt routine
into	call to interrupt routine if overflow
iret	return from interrupt routine

PROCESOR INTEL 80 286

- procesor s pokročilou architekturou podporující práci ve dvou režimech

- reálný režim – je plně kompatibilní s I 8086 což znamená, že je možno spouštět programy určené původně pro 8086 bez jakékoli modifikace

- chráněný režim – poskytuje vlastnosti podporující zpracování více úloh najednou

- dovoluje adresovat až 16 MB reálné paměti a až 1 GB virtuální paměti (adresová sběrnice je 24 bitová)

ARCHITEKTURA:

- procesor je složen ze čtyř nezávislých paralelně pracujících jednotek

- sběrnicová jednotka – zajišťuje přístup k reálné paměti – vybírá data, která ukládá do fronty dlouhé šest slabik (BU)

- instrukční jednotka – dekóduje vybrané instrukce a ukládá je do fronty (IU)

- prováděcí jednotka – realizuje dekódované instrukce (EU)

- adresová jednotka – je správcem paměti. Zajišťuje ochranu paměti a přepočítává virtuální adresy na reálné

REGISTRY:

- struktura registrů 8086 zůstala zachována. Přibyl 16 bitový registr MSW (machine status word) – jsou použity pouze spodní čtyři bity – PE, MP, EM, TS. Registr příznaků byl rozšířen o bity IOPC = 13,12 : určuje úroveň oprávnění pro operace V/V a bit NT = 14 : určuje práce instrukce IRET

OCHRANA PAMĚTI:

- pracuje pouze v chráněném režimu. Nový rys procesoru. Paměť je rozdělena na segmenty.

segment je souvislý paměťový prostor o délce až 64 KB. Každý segment je určen těmito parametry:

- báze segmentu (adresa začátku)

- limit segmentu (délku ve slabikách)

- přístupové práva

- čtyři úrovně oprávnění což umožňuje vyjádřit množství „důvěry“ poskytnuté určitému procesoru (0-3).

- existují také 4 typy segmentu:

- pouze provádět
- pouze číst
- provádět a číst
- číst i psát

INSTRUČNÍ SOUBOR:

- přidány privilegované instrukce (cca šestnáct instrukcí), některé staré byly upraveny

- nejvyšší priorita :

- LGDT, CIDT, LLDT, LTR, LMSN, CLTS, MCT

- operace vstupu a výstupu patří v chráněném módu mezi „částečně“ privilegované instrukce (smí použít „důvěryhodný“ proces – nesmí být tedy procesorem privilegovaným)

PROCESOR INTEL 80 386

- procesor s 32bitovou architekturou podporující práci ve třech režimech

- reálný režim – v tomto režimu je plně kompatibilní s I 8086 což znamená, že je možnou spouštět programy původně pro 8086 bez jakékoli modifikace (bez rekompilace)

- chráněný režim – je pro procesor nativní a je plně slučitelný s I 80 286.

- virtuální režim V86 – spustitelný v chráněném módu, dovoluje provozovat několik virtuálních strojů I 8086 na jednom procesoru 80 386

- může adresovat až 4 GB reálné paměti a až 64 TB virtuální paměti.

- ZÁKLADNÍ VYLEPŠENÍ:

- pracuje s 32 bitovými operandy (dvojslovo)
- segmenty mohou mít velikost až 4 GB
- má podporu pro stránkovací mechanizmus pamětí

- procesor se vyráběl ve dvou variantách
  - DX
  - SX – liší se od DX tím, že má obvody pro styk s okolím pouze 16bitové

ARCHITEKTURA:

- procesor je složen ze 6 nezávislých paralelně pracujících jednotek

- sběrnicová jednotka – realizuje styk s okolí, organizuje činnost sběrnice
- jednotka předvýběru – získává z paměti slabiky a sestavuje instrukce do fronty
- instrukční jednotka – přivádí instrukce na mikroinstrukce a ty ukládá do fronty
- prováděcí jednotka – zpracovává mikroinstrukce
- segmentační jednotka – převádí logické adresy na lineární (případně na fyzické)
- stránkovací jednotka – transformuje lineární adresu na fyzickou

REGISTRY

- struktura registrů I 8086 zůstala v zásadě zachována.

- 6 16 bitových segmentových registrů (CS, DS, SS, ES, FS, GS)
- registry indexové a pro všeobecné použití jsou 32 bitové a mohou být užity jako 8,16 nebo 32 bitové
- skupina registrů systémových adres (GDTR, LDTR, IDTR, TR) je stejná jako na I 80 286, jen báze byla rozšířena na 32 bitů (LDTR a TR má selektor 16 bitů)
- přibyly 32 bitové řídící registry CRO, CR1, CR2, CR3 (CRO = MSW)
- registr příznaků je 32 bitový a byl rozšířen o bity
  RF = 16 – maskuje ladící přerušení
  VM = 17 – zapíná režim V86
- nové 32bitové ladící registry DR0, DR1, DR2, DR3, které slouží pro uložení lineárních adres ladicích bodů
- registr DR6 je stavový a DR7 příkazový
- pro testování a ladění mechanizmu stránkování jsou určeny 32bitové registry TR6 a TR7

PROCESOR 80 486

- vyladěné jádro 386, ke kterému byla integrovaná aritmetická jednotka pro práci s čísly v plovoucí řádové čárce a interní cache o velikosti 8 kB. Tato cache je společná pro data i instrukce a může být doplněna vnější pamětí cache. Maximální pracovní frekvence byla 50 MHz. Instrukce jsou do značného stupně prováděny proudově – procesor je schopen provádět některé instrukce v jednom cyklu (především často používané instrukce). Procesor se vyráběl v několika variantách. Procesor 80 486 SX se lišil od základní varianty tím, že nemá funkční jednotku pro práci s čísly v pohyblivé řádové čárce. Dalším vylepšením bylo násobení vnitřní frekvence procesoru (over drive). Verze DX 2 pracuje vnitřně s dvojnásobkem vnější frekvence (25,33 MHz), verze DX se čtyřnásobkem. Zdvojením taktovací frekvence stoupne výkon procesoru asi o 50%. Dalším rozdílem bylo snížené napájecí napětí (3,3V)

ARCHITEKTURA:

IA – 32. Plně dvaatřicetibitový procesor. Pro zvýšení výkonu byly implementovány některé rysy architektury RISC. Instrukční sada tímto faktem nedotčena.

REGISTRY:

- struktura registrů stejná jako u 80 386

- registry pro všeobecné použití a indexové (EAX, EBX, ECX, EDX, ESI, EAI, ESP, EBP)
- segmentové registry – celkem šest (CS, SS, DS, ES, FS, GS)
- čítač instrukcí a registry příznaků (EIP, EFLAGS)
- řídící registry (CR0, CR1, CR2, CR3 (CRO=MSW)
- registry systémových adres (GDTR, LDTR, IDTR, TR)
- ladící registry (DR0, DR1, DR2, DR3, DR6, DR7)
- testovací registry – (TR3, TR4, TR5), pro testování paměti cache a pro ladění stránkování paměti (TR6 a TR7)

- registr příznaků je 32bitový a byl rozšířen o bit AC=18: zapíná generování přerušení při odkazu na objekt, který není zarovnán na příslušnou hranici (například zápis nebo čtení 16bitového slova na liché adrese)

ARITMETICKÉ OPERACE

1, SČÍTÁNÍ (add)

aaa	ASCI	adjust for addition
adc	destination source	add with carry
add		add bytes or word
daa		dec.adjust for addiction
inc	destination	increment

2, ODEČÍTÁNÍ (substrad)

cmp	destination source	compare
dec	destination	dekrement
neg	destination	negate
sbs	dest.,source	substrad with borrow
sub	dest, source	substrad

3, NÁSOBENÍ (multiply)

imul	source	integer multiply
mut	source	multiply

4, DĚLENÍ (divide)

cbw	court byte to word
cwd	court word to double word
div source	divide
i div source	integer divide

JEDNOTKA OPERACÍ V POHYBLIVÉ ŘADOVÉ ČÁRCE

- původně byla jednotka realizována jako samostatný čip (matematický koprocesor 8087, 80 287, 80 387)

- od procesoru 80 486 se stává součástí procesoru (FPU). Způsob ovládání (programování) zůstal nezměněn

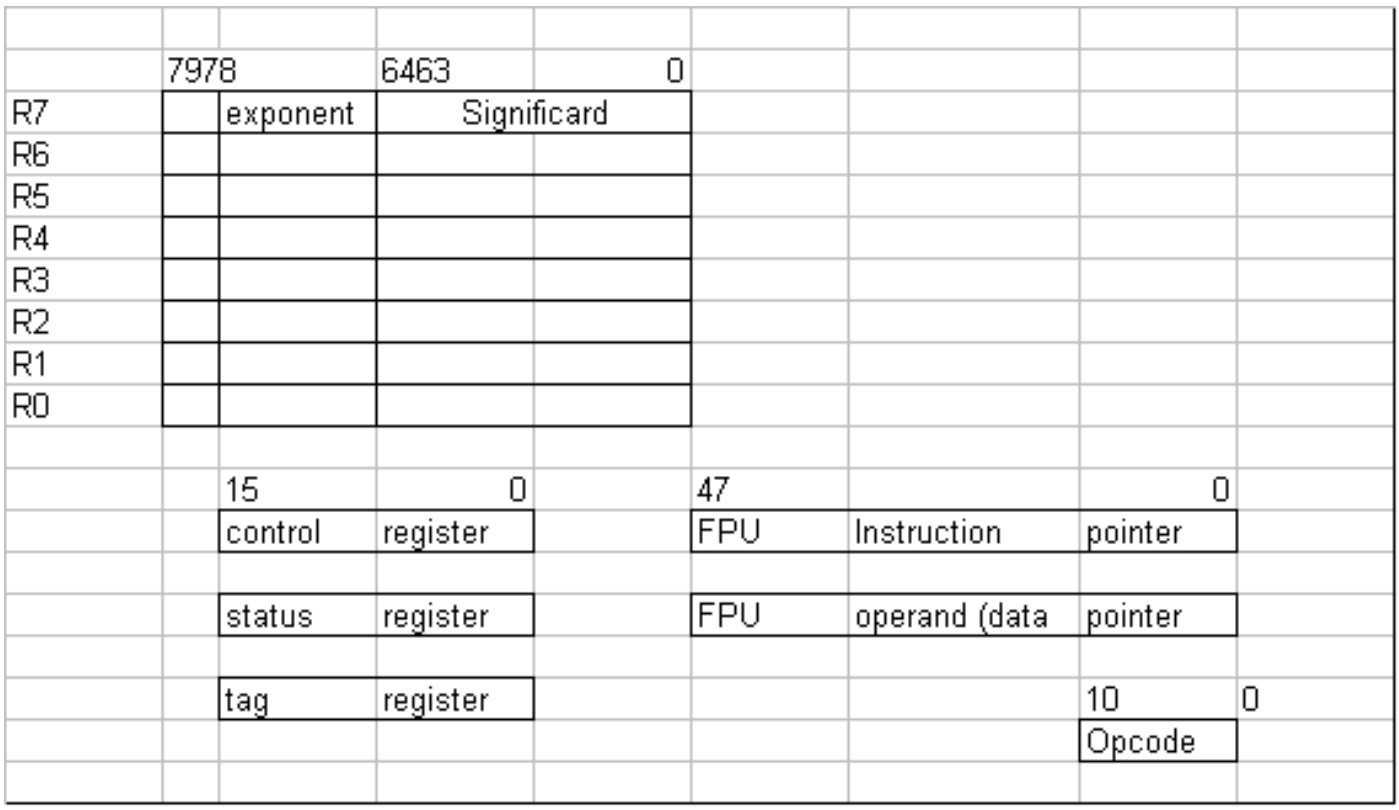
- jednotka interně pracuje výhradně s reálnými čísly o délce 80 bitů ve standardu IEEE – 754 a 854. Numerické výpočty prováděné jednotkou jsou nejméně o řád rychlejší než softwarová emulace s odpovídající přesností.

- jednotka operací v pohyblivé řadové čárce pracuje nezávisle na jednotce pro práci s celými čísly. Instrukce pro dekódování je předána jedné či druhé jednotce – výpočet se provádí paralelně. Pro úplnost nutno dodat, že v závislosti na procesoru, se mění počet jednotek. Procesor 80 486 má jednu IU a jednu FPU. Pentium má dvě IU a jednu FPU a Pentium Pro jednotky 2+2. Dosahuje se tak vyššího výpočetního výkonu – z hlediska programování procesoru se však nic nemění

jednotka umí pracovat s následujícími datovými položkami :

- celá čísla se znaménkem (dvojkový doplněk) o délce 16,32 a 64 bitů
- reálná čísla o délce 32 a 64 bitů a BCD čísla se znakem ve zhuštěném formátu o délce 18
  číslic

- uvedené typy čísel jsou na vstupu automaticky konvergovány do vnitřního zobrazení a při výstupu opět přivedeny na požadovaný tvar. Je-li to žádoucí je možný i výstup dat ve vnitřním formátu (80ti bitová položka --- se zvýšenou přesností)



- jednotka pro práci s reálnými čísly obsahuje 6 služebních a 8 datových registrů. Datové registry lze použít samostatně (jednotlivé a adresovatelné registry, 1-7) nebo jako zásobník – záleží na vykonávané instrukci

-stavový registr charakterizuje stav jednotky – obsahuje bity příznaků a bity signalizující chybové stavy. Bity 11 až 13 určují, který datový registr je vrchol zásobníku

- doplňující registr (tag) upřesňuje obsah každého datového registru (8 x 2 bity) – platný, nula, speciální (nečíslo), prázdný

-řídící registr plní 2 funkce :

- umožňuje nastavit některé parametry FPU – přesnost výpočtu, způsob zaokrouhlování (2 x 2 bity) a případně maskovat některé chybové stavy (ztráta přesnosti, podtečení, přetečení – celkem 6 možností)

- dva 48 bitové registry ukazatelů ukazují (ofset, segment) na poslední neřídící instrukci a její data – slouží k řešení chybových stavů. Registr operačního kódu obsahuje prvé 2 bajty poslední neřídící instrukce – podobně jako předchozí, poslouží při řešení chybových stavů.

JEDNOTKA OPERACÍ V POHYBLIVÉ ŘÁDOVÉ ČÁRCE

- INSTRUKČNÍ SOUBOR – MANIPULACE S DATY

prostý přesun

FLD	- load real
FST	- store real
FSTP	- store real and pop
FILD	- load integer
FIST	- store integer
FISTP	- store integer and pop
FBLD	- load BCD
FBSTP	- store BCD and pop
FXCH	- exchange registers

podmíněný přesun

FCMOVE	move if equal
FCMOVNE	move if not equal
FCMOVB	move if below
FCMOVBE	move if below or equal
FCMOVNB	move if not below
FCMOVNBE	move if not below or equal
FCMOVU	move if unordered
FCMOVNV	move in not unoredered

INSTRUKČNÍ SOUBOR – ARITMETICKÉ INSTRUKCE

základní

FADD	add real
FADDP	

doplňující

FABS	absolute value
FCHS	change sign
FRNDINT	round to integer
FSCALE	
FSQRT	square root
FXTRACT	extract exponent and significand

INSTRUKČNÍ SOUBOR – ŘÍDÍCÍ INSTRUKCE

FINCSTP	increment FPU register stack pointer
FFREE	Free floating – point registr
FCLEX	clear exception flags after checking for error conditions
FNCLEX	clear exception flags without checking for error conditions
WAIT	wait for FPU

INSTRUKČNÍ SOUBOR

funkce goniometrické

FSIN	sine
FCOS	cosine
FSINCOS	sine and cosine
FPTAN	pertial tangent
FPATAN	partial arctangent

funkce transcendentní

FYL2X	y.log 2x
FYL2XP1	y.log2(x+1)

funkce ostatní

F2XM1	2x-1

Computation

dat product = (5.6 x 2.4) + (3.8 x 10.3)

1, instrukce FLD VALUE1 uloží na vrchol zásobníku FPU [ST(0)] hodnotu prvé proměnné [5.6]. Ukazatel vrcholu zásobníku ukazuje na datový registr číslo 4

2, instrukce FMUL VALUE2 násobí hodnotu na vrcholu zásobníku [ST(0)] hodnotou druhé proměnné [2.4], která je uložena v paměti. Výsledek je na vrcholu zásobníku

3, instrukce FLD VALUE3 uloží na vrchol zásobníku hodnotu třetí proměnné [3.8]

4, instrukce FMUL VALUE4 násobí hodnotu na vrcholu zásobníku hodnotou čtvrté proměnné [10.3]. Výsledek je opět na vrcholu zásobníku

5, instrukce FADD(1) sečte hodnotu na vrcholu zásobníku a obsah registru ST(1), což je fyzicky registr R4. Výsledek je opět na vrcholu zásobníku

V případě, že struktura uložených hodnot nedovoluje provést požadovanou operaci přímo, jo možné instrukcí FXCH prohodit obsahy registrů zásobníků. Některé instrukce existují i v reverzní verzi (např. FDIV – ST(1)/ST(0) a FDIVR – ST(6)/ST(1)

JEDNOTKA NUMERICKÉHO PROCESORU – MMX

- vylepšením 32 bitové architektury procesorů Intel je rozšíření MMX. Účelem inovace je hardwarová podpora multimediálních a komunikačních aplikací (2D a 3D grafika, video, syntéza řeči, komprese a dekomprese datových toků při zpracování obrazu či zvuku) Rozšíření je založena na architektuře SIMD (single-instruction, multiple data). Podstatou je paralelní vykonání jedné instrukce nad několika datovými položkami (tedy současné). Rozšíření MMX přináší nové registry, nové datové formáty a rozšiřující instrukční sadu. To vše je včleněno do stávající architektury (IA-32) tak, že je bezezbytku zachována zpětná kompatibilita procesorů Intel. Nové aplikace mohou rozšíření MMX využívat, dříve vyrobené aplikace o něm prostě neví a budou proto fungovat tak, jako předtím.

- registrová sada MMX je tvořena osmi 64-bitovými registry, které se označují jako MM0-MM7. Mohou být použity pouze instrukcemi sady MMX (to znamená, že v nich mohou být uloženy jen pouze datové položky ve formátu MMX). Ačkoli to vypadá, že to jsou zvláštní registry, ve skutečnosti (fyzicky) se jedná o datové registry (R0-R7) numerického procesoru (označení MM je jen alias). Jsou definovány čtyři nové datové typy, každý o délce 64 bitů = sbalené bajty (8 bajtů), sbalená slova (čtyři slova), sbalená dvojslova (dvě slova) a prostá 64-bitová položka. Manipuluje s es nimi jako s celkem

- instrukční soubor má 57 instrukcí, které lze rozdělit do 7 skupin: přesun datových položek, operace srovnání, konverze dat, logické operace, posuny, aritmetické operace a instrukce vyprázdnění registrů systému MMX

Category	Wraparound	Signed Saturation	Unsigned Saturation
Aritmetic - addition	PADDB, PADDW	PADDSB	PADDUSB
	PADDD	PADDSW	PADDUSW
- substraction	PSUBB, PSUBW	PSUBSB	PSUBUSB
	PSUBD	PSUBSW	PSUBUSW
- multiplication	PMULL, PMULM		
- multiply and add	PMADD		
Comparsion – compare for equal	PCMPEQB		
	PCMPEQW		
	PCMPEQD		
- compare for	PCMPGTPB		
- greater than	PCMPGTPW		
	PCMPGTPD		
Conversion - pack		PACKSSWB	PACKUSWB
		PACKSSDW	
- unpack height	PUNPCKHBW		
	PUNPCKHWD		
	PUNPCKHDQ		
- unpack low	PUNPCKLBW		
	PUNPCKLKWD		
	PUNPCKLDQ		

Logical - and			
- and not			
- or			
- exclusive or			
Shift – shift left logical			
– shift right logical			
– shift right			
- arithmetic			
Data transfer - register to register			

POZNÁMKY

- SS a SP se nikdy nerovnají
- když je SP plný bude ukazovat na 0, největší FFFF
- když do SP uložíme AX jeho obsah se zmenší o 2, protože je 16bitový, SS se nemění
- instrukce skoku:
    - s podmínkou
    - bez podmínky
        - skok na blízko
        - skok na dálku

- při instrukce přesunu dat se obsah DS nebo ES změní, CS se nezmění
    ES, BX, AXD1 - ES – zaládujeme
    - BX – zaládujeme (ofset)

- při skokové instrukci se obsah CS změní
- instrukce se do zásobníku nikdy neukládají

- aritmetické instrukce
    - sčítání – přenos
    - odčítání – výpůjčka
    - bin.čísel – číslo je buď 8bitové nebo 16bitové
    - BCD

- řídit zaokrouhlováním
    - aritmeticky
    - nahoru
    - dolů
    - odseknutí

- 3 registry, které nám to dovolí řídit
- 8 80bitových datových registrů
- 16bitový registr status,řídící, doplňující (tag)
    - obsah registru je :
        - platný
        - nečíslo
        - prázdný
        - 0

- status – signalizace
        - dělení 0
        - není normalizované
        - ztráta přesnosti
        - přetečení
        - podtečení
        - nepovolená operace (0*nekonečno)
    - 3 bity
        - bit easy – procesor je zaneprázdněn

řídící –

- nastavením příslušného bitu na 1 zamaskuje chybové stavy
- 2 bity
- přesnost výpočtu
- zaokrouhlování