Inštrukčne orientovaná architektúra

Obsah prednášky

- opis základnej architektúry (ako funguje klasická CPU)
- 8 vs 32-bitové CPU, porovnanie a vplyv na formát inštrukcií
- súbor inštrukcií (8051 ako príklad), ich formát a rozdelenie
  - prenos
  - logické
  - aritmetické
  - bitové manipulácie
  - ...
- adresovacie módy
- programovací model

Základný cieľ:

- pochopiť činnosť inštrukčne orientovanej CPU počas vykonávania programu
- pochopiť činnosť a význam zásobníka v inštrukčne orientovanej CPU
- pochopiť súvislosti medzi HW a SW (štartovací kód, inicializácia zásobníka, asembler vs vyžší programovací jazyk, ...)

Vizualizácia jednoduchej centrálnej procesorovej jednotky (CPU)

https://courses.cs.vt.edu/csonline/MachineArchitecture/Lessons/CPU/index.html

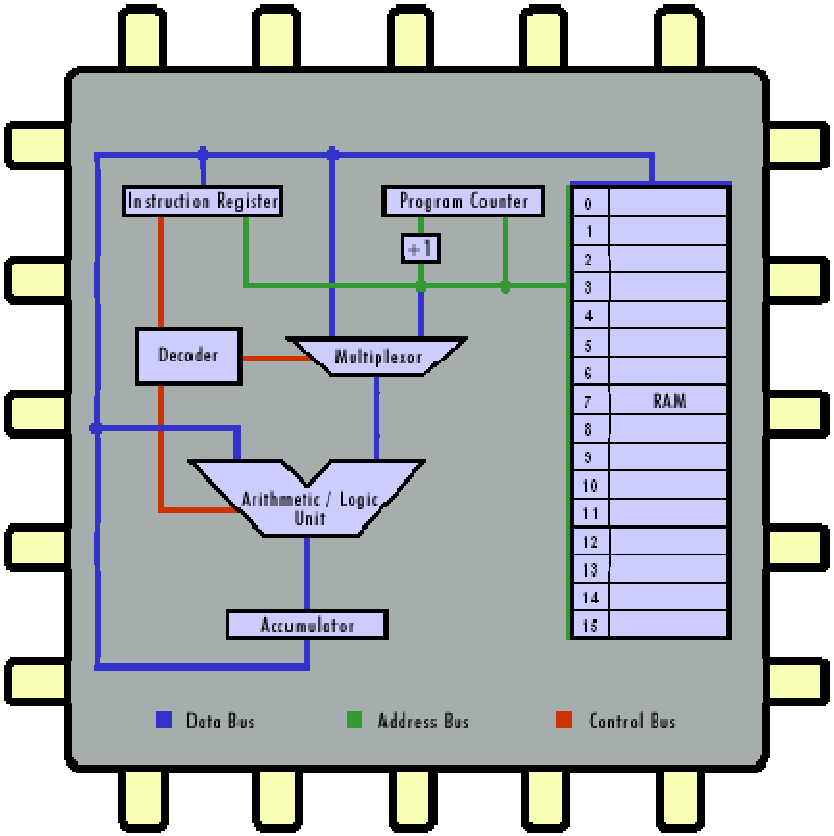
CPU obsahuje
- registre (IR-Instruction Register, PC-Program Counter, Accumulator)
- zbernice (adresová, dátová, riadiaca)
- aritmeticko-logickú jednotku (ALU – Arithmetic-Logic Unit)
- riadiacu jednotku (CU – Control Unit)

CPU je pomocou zberníc prepojená s **pamäťou** (pamäť programu, pamäť dát).

Pamäť programu obsahuje **inštrukcie**. Inštrukcie určujú akú činnosť (**operácie**) CPU vykonáva nad **dátami** uloženými v pamäti dát.

Programová a dátová pamäť môžu byť oddelené (využívajú odlišné adresové a dátové zbernice) – **harvardská architektúra**, alebo sú uložené v rovnakej pamäti (samozrejme na odlišných pozíciách) – von **Neumanova architektúra**.

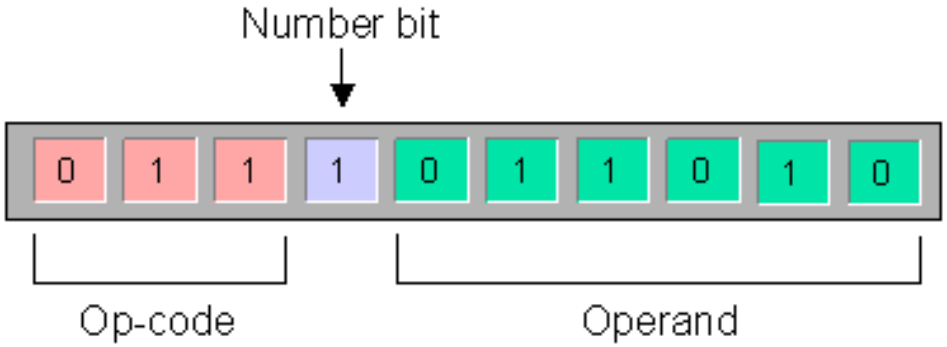
Jednoduchá CPU využitá pri vizualizácii



CPU inštrukcie

CPU načítava resp. vyberá (**fetch**), dekóduje (**decode**) a vykoná (**execute**) inštrukcie uložené v pamäti.

Formát inštrukcií výrazne závisí na architektúre CPU (8 vs 32-bitové) a tiež na type inštrukcií (aritmetické, logické, riadiace, ...).



Formát inštrukcií jednoduchej CPU (ak *Number bit* =1 O**perand** reprezentuje **číslo**, inak reprezentuje **adresu**).

Použité operačné kódy (Op-codes) a ich význam (asembler)

Op-code	*Mnemonic*	*Function*	*Example*
001	LOAD	Load the value of the operand into the Accumulator	LOAD 10
010	STORE	Store the value of the Accumulator at the address specified by the operand	STORE 8
011	ADD	Add the value of the operand to the Accumulator	ADD #5
100	SUB	Subtract the value of the operand from the Accumulator	SUB #1
101	EQUAL	If the value of the operand equals the value of the Accumulator, skip the next instruction	EQUAL #20
110	JUMP	Jump to a specified instruction by setting the Program Counter to the value of the operand	JUMP 6
111	HALT	Stop execution	HALT

Príklad programu sum na súčet dvoch čísel

#	*Machine code*	*Assembly code*	*Description*
0	001 1 000010	LOAD #2	Load the value 2 into the Accumulator
1	010 0 001101	STORE 13	Store the value of the Accumulator in memory location 13
2	001 1 000101	LOAD #5	Load the value 5 into the Accumulator
3	010 0 001110	STORE 14	Store the value of the Accumulator in memory location 14
4	001 0 001101	LOAD 13	Load the value of memory location 13 into the Accumulator
5	011 0 001110	ADD 14	Add the value of memory location 14 to the Accumulator
6	010 0 001111	STORE 15	Store the value of the Accumulator in memory location 15
7	111 0 000000	HALT	Stop execution

Pozri animáciu a sleduj:
- začiatok programu realizovaný vynulovaním PC a spustením načítavanie operačných kódov do registra IR
- postinkrementáciu PC po každej vykonávanej inštrukcii (nie je použitá inštrukcia skoku)

Príklad programu count na súčet zvoleného počtu čísel

#	*Machine code*	*Assembly code*	*Description*
0	001 1 000101	LOAD #5	These two operations set the count value to five
1	010 0 001111	STORE 15	
2	001 1 000000	LOAD #0	Initialize the count to zero
3	101 0 001111	EQUAL 15	Test to see if count is complete; if yes, skip next instruction and go to instruction 5; if no, go to next instruction
4	110 1 000110	JUMP #6	Set Program Counter to 6
5	111 0 000000	HALT	Stop execution
6	011 1 000001	ADD #1	Increment the count in the Accumulator
7	110 1 000011	JUMP #3	Set Program Count to 3

Pozri animáciu a sleduj:
- modifikáciu registra PC počas vykonávanie inštrukcie JUMP #6

Mikrokontrolér Intel 8051

Motivácia výberu CPU pre výučbu
- **jednoduchá** 8-bitová architektúra
- pomerne **jednoduchý súbor** inštrukcií
- **dostupný kvalitný simulátor** (Keil uVision)

Všeobecné vlastnosti **8-bitovej** CPU Intel 8051
- neortogonálny súbor inštrukcií (dominantné sú inštrukcie pre prácu s **akumulátorom**) – hlavný motív, malý počet dostupných formátov inštrukcií
- pomerne komplikované členenie **pamäťového modelu** (viacero typov pamätí)
- stále aktívne využívané jadro CPU vo vstavaných aplikáciách (využívané 10-kami výrobcov)

Všeobecné vlastnosti **32-bitovej** CPU (napr. s jadrami ARM)
- **ortogonálny** súbor inštrukcií (využitý rovnocenný registrový súbor namiesto špecifického akumulátora)
- typicky **lineárne** členený pamäťový priestor
- **dominantný** v moderných vstavaných zariadeniach

MCU 8051

Pomerne detailný HW opis MCU Intel 8051
http://matlab.fei.tuke.sk/subjects/jmvr/subory/podklady/Architektura_JM_8051.pdf
Všeobecný opis
http://www.dhservis.cz/popis8051.htm
Programátorský model a inštrukcie
http://www.fm.tul.cz/cip/download/instrukcnisada.pdf

KEMT podklady:
https://data.kemt.fei.tuke.sk/MikroprocesorovaTechnika/_web/wwwfiles/str%2004.htm
https://data.kemt.fei.tuke.sk/MikroprocesorovaTechnika/_web/
https://data.kemt.fei.tuke.sk/Architektury_pocitacovych_systemov/_materialy/prednasky/aps_cinnost_zasobnika.zip