CS241 Lec2- Machine Language

Graham Cooper

May 6th, 2015

Computer programs operate on data.
Computer programs are data
- von Neuman architecture
- reside in the same memory as the data they operate on.
∴ possible to write programs that manipulate other programs

How do we know what is program and what is data?
We don't

What do instructions look like? What instructions are there?
- Many different machine languages (processor specific)

For us: MIPS (simplified:) 18 different 32-bit instructions

The MIPS Machine

CPU: central Processing unit ALU: Arithmetic/Logic Unit, does math
Control Unit:
- decodes instructions
- dispatches to other parts of the computer to carry them out
Memory: Many kinds:
FAST
CPU, Cache, Main Memory (RAM), disk memory, network memory SLOW

Our focus is CPU and Main Memory
On the CPU - small amount of very fast memory (registers)
MIPS - 32 "general purpose" registers, $0 ... $31
- each holds 32 bits - 1 word

CPU can only operatoe on data in registers
$0 is always 0.
$31 is also special (later)
$30 is also sort of special (later)

eg. resiter operation: add the contents of regs $s + $t, put the result in $d

How many bits does it take to encode a reg #?
$2^5 = 32, \therefore 5$
$\therefore$ 15 bits to encode 3 reg #s
leaves 15 bits to encode the operation.

RAM: - Large amount of memory away from the CPU
- data travels between CPU + RAM on the BUS
- big array of n bytes $n \approx 10^{9+}$
- each cell has an address 0, 1, ..., n -1
- each 4 byte block is a word $\therefore$ words have addresses that are multiples of 4: 0, 4, 8, c, 10, 14, 18, 1c, 20, ...
- much slower than regs
- data in RAM must be transfered to regs before it can be used.

Communicating with RAM - 2 commands:

1. load
    - (Transfer a word from RAM to reg)
    - desired address goes in Memory Address Register (MAR)
    - goes out on the BUS
    - data at that location comes back on the BUS and goes into the Memory Data Register (MDR)
    - Value in MDR moved to destination register.
2. Store (reverse of load)

The computer doesn't know which words contain instructions and which contain data.Then how does it execture code?
Special reg called PC (program counter) hods the address of the next instruction to run.
By convention, we guarantee that a specifc address (say, 0) contains code, initialize PC = 0.

Computer then runs the fetch-execute cycle:

- $PC \Leftarrow 0$
- loop
  - IR $\leftarrow$ MEM[PC] (IR = instruction register, holds the current instruction)
  - PC $\leftarrow$ PC + 4
  - decode and execute instruction in IR
- end loop
- this is the only program the machine really runs

NOTE: PC holds the address of the next instruction, while the current instruction is running

Q: How does the program get executed?
A: A program called a loader that puts your program in memory and sets PC to the address of the first instruction in your program.

Q: What happens when your program ends?
A: Need to return control to the loader. Need to set PC to the address of the next instruction in the loader. Which address is that? $31 will contain the correct address. Set PC to register $31. jr instruction in mips.

Example 1: Add the value in $5 to the value in $7, store the result in $3 and return.

MIPS MACHINE CODE

Location	Binary	Hexadecimal	Meaning
00000000	0000 0000 1010 0111 0001 1000 0010 0000	0x00A71820	add $3 $5 $7
00000004	0000 0011 1110 0000 0000 0000 0000 1000	0x03e00008	jr $31

Example 2: Add 42 + 52, store in $3, return.
lis $d - "Load immediate and skip"
- treat the next word as a value, load into $d, and skip over it to the following instruction.

MIPS MACHINE CODE

Location	Binary	Hexadecimal	Meaning
00000000	0000 0000 1010 0111 0001 1000 0010 0000	0x00A71820	add $3 $5 $7
00000004	0000 0011 1110 0000 0000 0000 0000 1000	0x03e00008	jr $31