Computer Science 210

Computer Systems 1

2007 Semester 1

Lecture Notes

The Programmer's View of Computer Hardware

James Goodman

Department of Computer Science

Who Am I?

- Prof. James Goodman
- Computer Science Department
- Science Centre 303-591, 38 Princes St., City
- Office Hours: No scheduled time: drop by my office or make appointment
- E-mail: goodman@cs.auckland.ac.nz

Recommended Readings

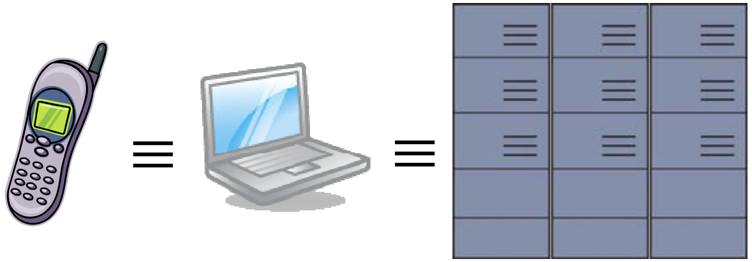
- These notes (only after the lecture):
  http://www.cs.auckland.ac.nz/compsci210s1t/lectures
- Dr. Bruce Hutton's lecture notes:
  http://www.cs.auckland.ac.nz/compsci210s1t/resources

Why Study Computer Organization?

- Understanding how hardware and software communicate will make you a better programmer
- Some things change; some things stay the same
  - Moore's Law vs. fundamental laws
- Appreciate the power of abstraction
  - Don't write in assembly language if you don't have to!
- "Real Programmers do it in Assembly"
  - No longer an important skill

All Computers are the Same!

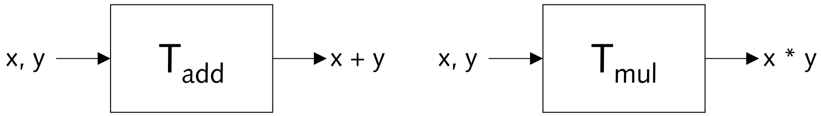
- All computers, given sufficient time and memory, can compute exactly the same things.



Cellphone Laptop Supercomputer

Turing Machine

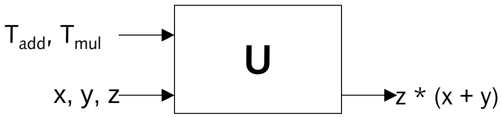
- Mathematical model of a device that can perform any computation – Alan Turing (1937)
  - ability to read/write symbols on an infinite "tape"
  - state transitions, based on current state and symbol
- Everything that can be computed can be performed by some Turing machine. *(Turing's thesis)*



Turing machine that adds *Turing machine that multiplies*

Universal Turing Machine

- Turing described a Turing machine that could implement all other Turing machines.
  - inputs: data, plus a description of computation (Turing machine)



Universal Turing Machine

U is programmable – so is a computer!

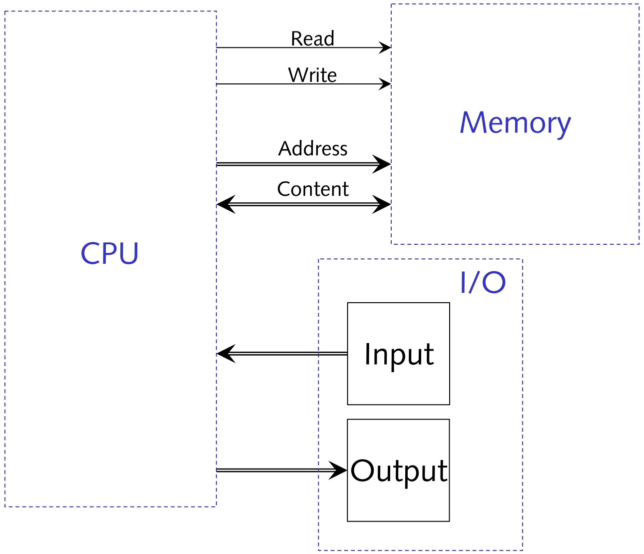
- instructions are part of the input data
- a computer can emulate a Universal Turing Machine, and vice versa

Therefore, a computer is a universal computing device!

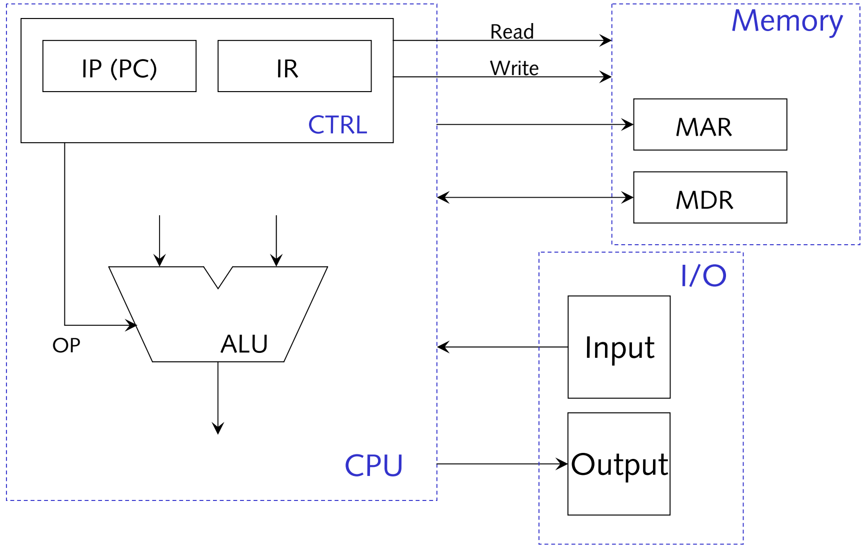
From Theory to Practice

- In theory, computer can ***compute*** anything that's possible to compute if you
  - Have enough memory
  - Can wait long enough
- In practice, ***solving problems*** involves computing under constraints.
  - Time: photoshop, weather forecast,...
  - Cost: hotel "key", PDA, ...
  - Power: cell phone, laptop, ...

The Von Neuman Computer



The Von Neuman Computer



The von Neuman Model

- Computer consists of CPU, Memory, I/O
- Memory may contain instructions or data (or meta-data)
- Does only one thing: the Instruction/Execution cycle

The Instruction/Execution Cycle

```
Do forever {
    Fetch instruction into IR from memory address in IP
    Update IP for next instruction
    Decode instruction
    Evaluate addresses
    Fetch operands from memory
    Store result
}
```

The Instruction/Execution Cycle:
Variant for Control Instructions

Do forever {
  Fetch instruction into IR from memory address in IP
  Update IP for next instruction
  Decode instruction
  Evaluate test criterion
  If success, store new address to PC
}

A Few Sample Instructions

Instruction	Meaning
add A, B, C	C = A + B
sub A, B, C	C = A – B
mul A, B, C	C = A * B
bne A, B, Label	if (A != B) goto Label
halt	?

- A *Label* designates a memory location.
- A Label can be either an instruction or a variable

A Simple Program

Instructions:

```
L1:   add   VA, VB, VA
L2:   sub   VC, VD, VC
L3:   mul   VC, VE, VE
L4:   bne   VA, VC, L1
L5:   halt
```

Initial values:

```
VA: 0
VB: 1
VC: 6
VD: 2
VE: 5

IP: L1
```