ECE222

Andy Zhang

Fall 2014

Contents

Chapter 1

Computers

1.1 Classes (1.1)

1.1.1 Personal

- Desktop, laptop, tablet, smartphones
- %1 of all CPUs sold(10 billion in 2008)
- Cost: $20 - 200

1.1.2 Embedded

- Integrated into a larger device or system
  - Automotive(airbags, ABS, ...)
  - Appliances: stove, microwave...
  - Airplanes
- 99% of all CPUs
- Cost: Microchip PIC12: $0.41

1.1.3 Servers

- Provides service to many users
  - Cloud computing (Amazon EC2, Azure ...)
  - Mainframes (IBM System Z) used by banks, universities, governments due to high reliability
  - Supercomputers — weather modelling, protein folding..
- <1% of all CPUs sold
- cost: $2000 / chip

1.2 Structure (1.2)

Definition : a computer is a ‘programmable device that can store, retrieve and process data’ — Merriam Webster

Computers of all classes can be decomposed into five types of functional events

1. Input: Mouse, Punchard, Touch Screen, Camera
2. Output: Printer, Screems.
3. Storage: Data, instructions (binary)
    - Memory is organized into a linear array of bytes
4. ALU: Arithmetic Logic Unit
    - Performs operations on data stored in registers
    - Add, multiply, AND, NOT, ...
5. Control Unit
    - Interpret instructions, fetch operands, control ALU

Chapter 2

Processors - September 11

2.1 Processor

PC program counter stores memory address of next instruction

IR instruction register stores instruction read from memory

MAR memory adderss to register outputs address to memory

MDR memory data register. Holds data/instructions from memory or going to memory

2.2 Instruction execution

2.2.1 Instruction Fetch(IF)

- Copy PC contents to MAR and assert R/W control signal
- Wait for response from memory and copy MDR contents to IR
- Increment PC

2.2.2 Instruction Decode(ID)

- Interpret bits in IR

2.2.3 Operand Fetch(OF)

- Read data from registers and/or extract constants from IR

2.2.4 Execute(EX)

- Use ALU or read memory(load) or write memory(store)

2.2.5 Writeback(WB)

Write result to a register

Eg Execute Load R2, LOC (memory address label)

1. Always same as above
2. Recognize "Load"
3. Etract LOC from IR
4. Copy LOC to MAR and assert R/W control signal
5. Copy MDR Contents to R2

2.2.6 Homework

```
ADD R4, R2, R3 ($R4 <- [R2] + [R3]$)
 Store R4, LOC
```

2.3 Design Paradigms

2.3.1 CISC

Complex Instruction Set Computer

- Machine instructions can perform complex operations

  E.g. (x86) *movsb* copies an array of bytes

- Instructions are variable length
- Operands come from registers or memory

  E.g *M68K* ADD DO, LOC (mem[LOC] <- [DO] + [mem[LOC])

- Complex addressing modes

  E.g. (M68K) ADD DO, (A0)+

- Smaller object code
- Direct support of High Level Language constructs
- Ease of assembly language programming
- Hardware is difficult to pipeline(speed up)