# CS250 Midterm 1 Review

Theo Park

September 21, 2022

## Contents

**1 Why Computer Architecture 2**
1.1 Definitions . . . 2
1.2 C Compiling Process . . . 3
1.3 Mechanical Computers . . . 3
1.4 Vacuum Tube Computers . . . 3
1.5 Transistor . . . 4
1.6 Two Architectures . . . 4

**2 Representation 4**
2.1 Electrical Representation of Bits . . . 4
2.2 Bit String . . . 4

**3 Regular Representations 5**
3.1 Unsigned integer, base 2, weighted positional . . . 5
3.2 Sign Magnitude . . . 5
3.3 Two's Complement . . . 5

**4 Casting/Sign Extension 5**

**5 Overflow 6**

**6 Gray Code 6**

**7 ASCII 6**
7.1 History . . . 6

**8 Order of Bytes in Memory 6**

**9 Floating Point Representation (IEEE 754) 6**
9.1 Exponent . . . 7
9.2 Mantissa . . . 7
9.3 Runtime Anomalies . . . 7

**10 Memory 7**
10.1 Pointing Function . . . 7
10.2 Register . . . 8

**11 Pointing 8**
11.1 Decoder . . . 8
11.2 Selecting bus . . . 8

**12 Processor 8**
12.1 Revisit of Architectures . . . 8
12.2 Processor is . . . 9

**13 General One-Step Processor Circuit 9**
13.1 Fetch-Execute Cycle . . . 9
13.2 Clock rate and Instruction rate . . . 10
13.3 Start and Stopping Hardware . . . 10

**14 Instruction Encoding 10**
14.1 Instruction Set Architecture . . . 10
14.2 Instruction Size . . . 11

**15 x86 11**

# 1 Why Computer Architecture

## 1.1 Definitions

- *Computer* is a machine that can be programmed to **carry out computation automatically**
- *Architecture* is a **conceiving, planning, and designing structures**
  - CA has purpose only when given SW
- *Software* is a **description of a computation** expressed in a programming language, any data, and documentation

  - Purpose 1: Definining an DS & A
  - Purpose 2: Executing

- *Interpreter* **executes software**
  - Directly executes instructions expressed in a PL
  - **Does NOT rely on "Turtles all the way down"** (interpreter for interpreter for interpreter...) approach

- *Compiling* is the process of **traslating** programs written in one **HLL** (High-level language) into a **LLL** that **has a machine interpreter**

## 1.2 C Compiling Process

// TODO Find a better way to put diagram in Org files

```
source_code -> preprocessor -> preprocessed source code -> compiler -> assembly code ->
```

- Preprocessed Source Code: Does not contain **comments, macros, includes**, etc
- Assembly Code: **Machine specific**

## 1.3 Mechanical Computers

- Antikythera Mechanism (200B.C): Count Olumpics days
- Charles Babbage (1849)

### 1.3.1 Disadvantages

- Parts are small, require individual assembly
- Part shape and size determine computational function
- Parts cause waer and accuracy degrades over time
- Algorithm are slow

## 1.4 Vacuum Tube Computers

- Colossus

#### 1.4.1 Disadvantages

- About the same volume as mechanical computer
- Uses a lot of electrical energy
- Vacuum tubes burn out

### 1.5 Transistor

- First one built at AT&T Bell Labs
- Used to use germanium crystal, now use silicon
- Futures are graphene or single layer of carbon

### 1.6 Two Architectures

#### 1.6.1 Harvard Architecture

**Separate memories** for instructions and data

#### 1.6.2 Von Neumann Architecture

**Single memory** for instruction and data

## 2 Representation

### 2.1 Electrical Representation of Bits

- **V (max) voltage V -** $\Delta$ is recognizes as 1
- **0 to 0** $+ \delta$ is recognizes as 0
- **Rising edge** and **falling edge** are ignored

### 2.2 Bit String

- **Bus: Collection of k wires carrying k-bits**
- **k-bits** on **k-wires**
- k-bits can represent up to $2^k$ **values**
- *Bit strings are only meaningful when it is paried with a representation*

## 3 Regular Representations

Unsigned and 2's complement integers are native data types for most modern circuits

### 3.1 Unsigned integer, base 2, weighted positional

**Regular binary number** that we think of normally.
`001011` $= 0 \times 2^5 + 0 \times 2^4 + 1 \times 2^3 + 0 \times 2^2 + 1 \times 2^1 + 1 \times 2^0 = 11$

### 3.2 Sign Magnitude

**UIB2WP but the MSB is the sign** (MSP = left most bit).
`101011` $= -1(0 \times 2^4 + 1 \times 2^3 + 0 \times 2^2 + 1 \times 2^1 + 1 \times 2^0) = -11$

#### 3.2.1 Characteristics of sign magnitude

- There are two zeros (0000 = +0, 1000 = -0)
- Less number can be represented (duh)

### 3.3 Two's Complement

**MSB weight is negative**
`101011` $= -(1 \times 2^4) + 1 \times 2^3 + 0 \times 2^2 + 1 \times 2^1 + 1 \times 2^0) = -5$

#### 3.3.1 Characteristics of two's complement

- Only one bit string for zero
- **Invert bit string and add 1 to get the negative**
- **Uses the same circuit as unsigned integer add/subtraction**

## 4 Casting/Sign Extension

- Unsigned integer: **Add 0 in front**
- 2's complement: **Add MSB in front**

## 5 Overflow

- Adding two k-bit unsigned integer resulting in (k+1)-bit result
- $A +_k B = (A + B) \bmod k$ prevents it

## 6 Gray Code

For sensors where bits need to be detected fast, "gray code" where only one bit changes per number is used.

## 7 ASCII

### 7.1 History

*Baudot Code* in 1870 used to represent $2^5$ characters with 5 keys.

#### 7.1.1 Design of ASCII

- **Designed for machine, not human**
- Alphabetic order = integer order of chracter codes
- Upper and lower case only differ in **bit 7, the MSB** (these days 8 bits are used, but MSB is always 0)

#### 7.1.2 Unicode

- Up to 4 bytes per character
- Currently 14.0, supports emoji

## 8 Order of Bytes in Memory

- *Big Endian*: **MSB comes first** `0x5060` is stored as `0x5060`
- *Lil Endian*: **LSB comes first** `0x5060` is stored as `0x6050`

## 9 Floating Point Representation (IEEE 754)

```
|S| Exponent | mantissa |
```

### 9.1 Exponent

Exponent is a biased integer. The initial range is -127 $<$ e $<$ 127, sign is made implicit by E = e + Bias = e + 127.

### 9.2 Mantissa

Unless the number is 0, the MSB of the mantissa must be 1 -> No need to store! (**hidden bit**). Instead, **one extra precision bit** is stored in the end.

### 9.3 Runtime Anomalies

1. $E = 0$, $Mantissa = 0$,: $\pm 0$, depending on the sign bit
2. $E = 0$, $Mantissa \neq 0$, $Mantissa\ MSB = 0$: De-normalized number, gradual underflow
3. $E = 255$, $Mantissa = 0$: $\pm\infty$; in general, overflow is set to infinity to help people
4. $E = 255$, $Mantissa \neq 0$: Not a Number

## 10 Memory

### 10.1 Pointing Function

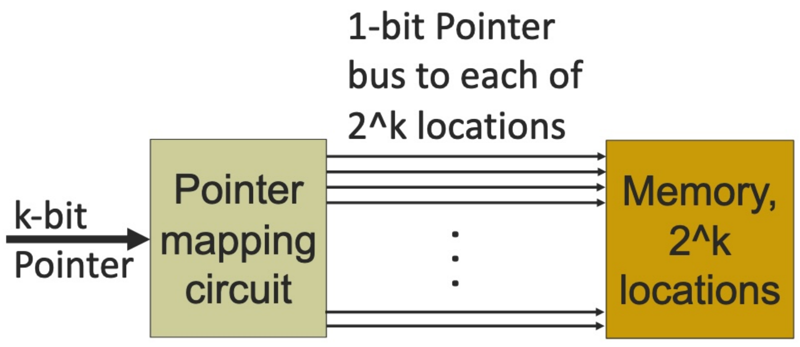

Figure 1: Abstract Pointing Function Diagram by Prof. George B Adams

Memory chips are 2D, $2^{k/2} \times 2^{k/2}$ grid creates $2^k$ intersections -> Only $2{*}2^{k/2}$ wires needed! Optimized design can reduce number of wires by $\sqrt{2^k}/2$

### 10.2 Register

k-bit register has k latches to store $2^k$ bit strings.

## 11 Pointing

### 11.1 Decoder

A circuit with n wires input and 2^n wires output. Decoding output is selected or not_selected

| X | Y | D0 | D1 | D2 | D3 |
|---|---|---|---|---|---|
| 0 | 0 | 1 | 0 | 0 | 0 |
| 1 | 0 | 0 | 1 | 0 | 0 |
| 0 | 1 | 0 | 0 | 1 | 0 |
| 1 | 1 | 0 | 0 | 0 | 1 |

2-to-4 decoder truth table. **n inputs and 2^n outputs**.

### 11.2 Selecting bus

- *Bus*: Group of n wires, carry n bit
- *Multiplexer (mux)*: Selects **from 2^n k-bit input buses, outputs to 1 k-bit output bus**
- *Demultiplexer (Demux)*: Reverse of mux

## 12 Processor

### 12.1 Revisit of Architectures

- Harvard: Optimized design and simultaneous access for data and instructions; storage inefficiency
- Von Neumann: Less memory needed; security issue

### 12.2 Processor is

- Not CPU
- Includes co-processor and microcontroller
- Building specific one purpose is expensive -> General purpose processor

## 13 General One-Step Processor Circuit

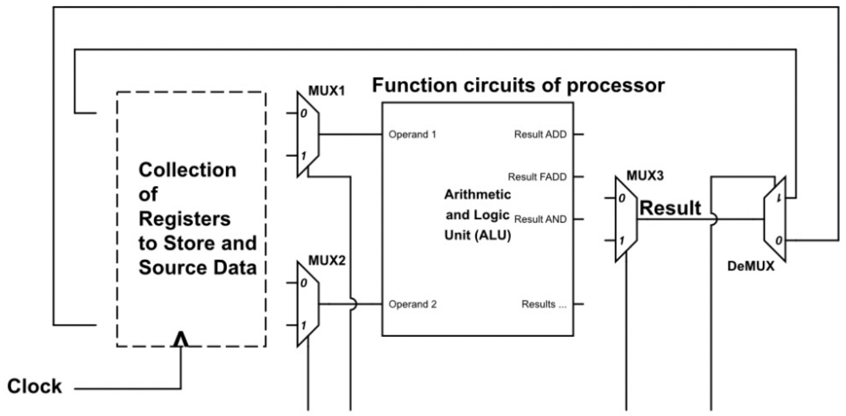
Function circuits of processor
MUX1
MUX2
MUX3
DeMUX
Collection of Registers to Store and Source Data
Clock
Operand 1
Operand 2
Arithmetic and Logic Unit (ALU)
Result ADD
Result FADD
Result AND
Results ...
Result
0
1

Figure 2: General One-Step Processor Circuit by Prof. George B Adams

Key points:

- Input goes from register collection to MUX1 and MUX2, they choose two k-bit strings
- ALU **continously** to the calculation and outputs to the MUX3
- MUX3 chooses one of the result and outputs
- DeMUX put the bit string back to **correct** location (MUX3 will mess it up if it puts it back to the collection w/o DeMUX)

### 13.1 Fetch-Execute Cycle

```
while (power is on) {
  fetch;
  execute;
}
```

Wow fancy algorithm.

## 13.2 Clock rate and Instruction rate

- Clock rate: Worst circuit propagation delay
- Registers and mux/demux delay exists
- ALU propagation delay varies a lot since it's where all the calculations happen

$$\text{CPU Time} = \frac{Instructions}{Program} \times \frac{ClockCycles}{Instruction} \times \frac{Seconds}{Clockcycle}$$

- Instruction per program: Software runtime dependent. 251 flashback
- Clock cycles per instruction: Compiler and circuit design dependent
- Seconds per clock cycle: Worst case propagation delay, the clock rate of the CPU

## 13.3 Start and Stopping Hardware

Hardware is designed to run 24/7. For your computer, there is an idle loop to run to continue fetch-execute cycle.

Bootstrap: **power-on reset** of latches to put the computer in known state to start fetch-execute cycle,

# 14 Instruction Encoding

## 14.1 Instruction Set Architecture

Set of operations chosen by a careful consideration to make processors the more expensive.

```
| opcode | operand 1 | operand 2 | ... | Result 1 | Result 2 | ...
```

- **Opcode**: Selects ALU result to use, contains number of operands
- **Operands**: Provide input to ALU
- **Results**: Pointers to storage locations

### 14.2 Instruction Size

#### 14.2.1 Variable Length

- Marketing people loves it
- Compiler people hates it
- Computer slow
- Complex

#### 14.2.2 Fixed Length

- Marketing people hates it
- Compiler people loves it
- Ez
- Instructions may not utilize all the bits

#### 14.2.3 What drives the length?

- Not opcode, $2^k \times 2 = 2^{k+1}$ only 1 bit is needed to double the operations
- Operands, results fields are the bad guys

#### 14.2.4 Fix Pointer Size

- Make pointer size a constant
- ALU circuits access memory, fetch operands and store result in fixed number of registers
- Main memory grow, pointer unchanged

## 15 x86

Intel rich.