

The Von Neumann Architecture

Odds and Ends

Chapter 5.1-5.2

CMPUT101 Introduction to Computing (c) Yngvi Björnsson & Vadim Bulitko 1

Designing Computers

- All computers more or less based on the same basic design, the Von Neumann Architecture!

CMPUT101 Introduction to Computing (c) Yngvi Björnsson & Vadim Bulitko 2

The Von Neumann Architecture

- Model for designing and building computers, based on the following three characteristics:
 - The computer consists of four main sub-systems:
 - Memory
 - ALU (Arithmetic/Logic Unit)
 - Control Unit
 - Input/Output System (I/O)
 - Program is stored in memory during execution.
 - Program instructions are executed sequentially.

CMPUT101 Introduction to Computing (c) Yngvi Björnsson & Vadim Bulitko 3

The Von Neumann Architecture

CMPUT101 Introduction to Computing (c) Yngvi Björnsson & Vadim Bulitko 4

Structure of the Memory Subsystem

- Fetch(address)**
 - Load address into MAR.
 - Decode the address in MAR.
 - Copy the content of memory cell with specified address into MDR.
- Store(address, value)**
 - Load the address into MAR.
 - Load the value into MDR.
 - Decode the address in MAR.
 - Copy the content of MDR to memory cell with the specified address.

CMPUT101 Introduction to Computing (c) Yngvi Björnsson & Vadim Bulitko 5

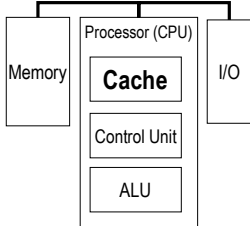
Implementation of the Memory Subsystem

| Address | Memory Cell |
|---------|-------------|
| 0000 | (0) |
| 0001 | (1) |
| 0010 | (2) |
| ... | ... |
| 1110 | (14) |
| 1111 | (15) |

CMPUT101 Introduction to Computing (c) Yngvi Björnsson & Vadim Bulitko

CACHE - Modern addition

- High-speed memory, integrated on the CPU
 - Ca. 10 times faster than RAM
 - Relatively small (128-256K)
- Stores data most recently used
 - Principle of Locality
- When CPU needs data:
 - First looks in the cache, only if not there, then fetch from RAM.
 - If cache full, new data overwrites older entries in cache.

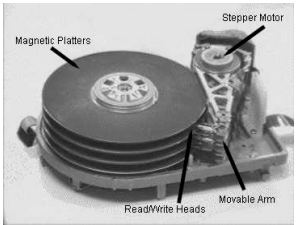


CMPUT101 Introduction to Computing (c) Yngvi Björnsson & Vadim Bulitko 7

I/O Subsystem: Hard-Drives

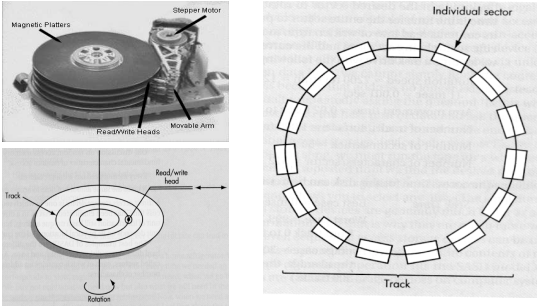
- Uses magnetic surfaces to store the data.
 - Each surface has many circular tracks.
 - Each track consists of many sectors.

The surfaces rotate at a high speed
Typically ~7000 rev/min
The read/write arm moves:
back and forth to locate a track



CMPUT101 Introduction to Computing (c) Yngvi Björnsson & Vadim Bulitko 8

Hard-Drive



CMPUT101 Introduction to Computing (c) Yngvi Björnsson & Vadim Bulitko 9

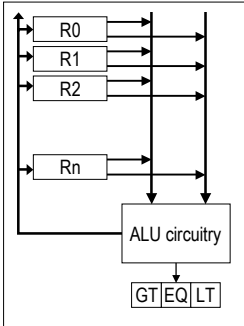
Disk Access Time

- The time it takes to read/write data to a disk, consists of:
 - Seek time
 - The time it takes to position the read/write head over correct track (depends on arm movement speed).
 - Latency
 - The time waiting for the beginning of the desired sector to get under the read/write head (depends on rotation speed)
 - Transfer time
 - The time needed for the sector to pass under the read/write head (depends on rotation speed)
- Disk Access Time = Seek time + Latency + Transfer time
- Measure worst, best, and average case. (Example: p. 189)

CMPUT101 Introduction to Computing (c) Yngvi Björnsson & Vadim Bulitko 10

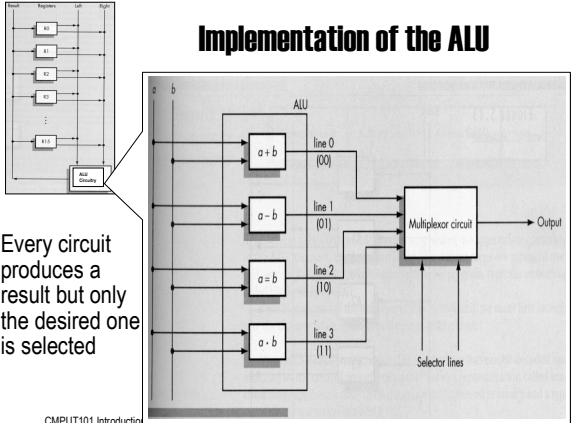
Structure of the ALU

- Registers:
 - Very fast local memory cells, that store operands of operations and intermediate results.
 - CCR (condition code register), a special purpose register that stores the result of <, =, > operations
- ALU circuitry:
 - Contains an array of circuits to do mathematical/logic operations.
- Bus:
 - Data path interconnecting the registers to the ALU circuitry.



CMPUT101 Introduction to Computing (c) Yngvi Björnsson & Vadim Bulitko 11

Implementation of the ALU



Every circuit produces a result but only the desired one is selected

CMPUT101 Introduction to Computing

Structure of the Control Unit

- PC (Program Counter):
 - stores the address of next instruction to fetch
- IR (Instruction Register):
 - stores the instruction fetched from memory
- Instruction Decoder:
 - Decodes instruction and activates necessary circuitry

The diagram shows a central horizontal line representing the bus. Below it, a box contains three components: a '+1' counter, a 'PC' (Program Counter), and an 'IR' (Instruction Register). The '+1' counter has an arrow pointing to the PC. The PC has an arrow pointing to the IR. Below the PC and IR is an 'Instruction Decoder' box. Arrows show the flow of information: from the bus to the PC, IR, and Instruction Decoder, and from the Instruction Decoder back to the bus.

CMPUT101 Introduction to Computing (c) Yngvi Björnsson & Vadim Bulitko 13

Machine Language Instructions

- A machine language instruction consists of:
 - Operation code, telling which operation to perform
 - Address field(s), telling the memory addresses of the values on which the operation works.
- Example: ADD X, Y (Add content of memory locations X and Y, and store back in memory location Y).
- Assume: opcode for ADD is 9, and addresses X=99, Y=100

| | | | |
|-----------------|---------------------|---------------------|--|
| Opcode (8 bits) | Address 1 (16 bits) | Address 2 (16 bits) | |
| 00001001 | 000000001100011 | 000000001100100 | |

CMPUT101 Introduction to Computing (c) Yngvi Björnsson & Vadim Bulitko 14

Implementation of the Control Unit

The diagram shows a block labeled 'Implementation of the Control Unit'. It features a table of Op Code and Instruction, and a diagram of an instruction decoder. The table lists: 000 (ADD), 001 (LOAD), 010 (JUMP), and 111 (HALT). The decoder diagram shows an 'Op Code' field with three bits (b, b, b) and an 'Address' field. Lines from the decoder are labeled: Line 000 (ADD) - Enable the ADD operation, Line 001 (LOAD) - Enable the LOAD operation, Line 010 (JUMP) - Enable the JUMP operation, and Line 111 (HALT) - Enable the HALT operation.

CMPUT101 Introduction to Computing (c) Yngvi Björnsson & Vadim Bulitko 15

von Neumann Architecture

The diagram illustrates the von Neumann Architecture. It is divided into four main sections: Memory unit, Arithmetic/logic unit, Input/output, and Control unit. The Memory unit contains MAR (Memory Address Register), MDR (Memory Data Register), Memory decoder circuits, and Random access memory. The Arithmetic/logic unit contains RO (Register 0), R1, R2, R3, and an ALU (Arithmetic Logic Unit). The Input/output section contains an I/O controller and an I/O device. The Control unit contains the PC (Program Counter), IR (Instruction Register), and Instruction decoder circuit. A '+1' counter is also shown. A bus connects all these units. A 'Fetch/store controller' is connected to the bus and provides an 'F/S signal'. 'Selector lines' connect the ALU to the 'Condition code register' (GT, EQ, LT). A small portrait of a man is shown in the bottom right corner.

CMPUT101 Introduction to Computing (c) Yngvi Björnsson & Vadim Bulitko 16

How does this all work together?

- Program Execution:
 - PC is set to the address where the first program instruction is stored in memory.
 - Repeat until HALT instruction or fatal error
 - Fetch instruction
 - Decode instruction
 - Execute instruction
 - End of loop

CMPUT101 Introduction to Computing (c) Yngvi Björnsson & Vadim Bulitko 17

Program Execution (cont.)

- Fetch phase
 - PC --> MAR (put address in PC into MAR)
 - Fetch signal (signal memory to fetch value into MDR)
 - MDR --> IR (move value to Instruction Register)
 - PC + 1 --> PC (Increase address in program counter)
- Decode Phase
 - IR -> Instruction decoder (decode instruction in IR)
 - Instruction decoder will then generate the signals to activate the circuitry to carry out the instruction

CMPUT101 Introduction to Computing (c) Yngvi Björnsson & Vadim Bulitko 18

Program Execution (cont.)

- Execute Phase
 - Differs from one instruction to the next.
- Example:
 - LOAD X (load value in addr. X into register)
 - IR_address -> MAR
 - Fetch signal
 - MDR -> R
 - ADD X
 - left as an exercise

CMPUT101 Introduction to Computing

(c) Yngvi Björnsson & Vadim Bulitko

19

Instruction Set for Our Von Neumann Machine

| Opcode | Operation | Meaning |
|--------|-------------|--|
| 0000 | LOAD X | CON(X) -> R |
| 0001 | STORE X | R -> CON(X) |
| 0010 | CLEAR X | 0 -> CON(X) |
| 0011 | ADD X | R + CON(X) -> R |
| 0100 | INCREMENT X | CON(X) + 1 -> CON(X) |
| 0101 | SUBTRACT X | R - CON(X) -> R |
| 0101 | DECREMENT X | CON(X) - 1 -> CON(X) |
| 0111 | COMPARE X | If CON(X) > R then GT = 1 else 0 If CON(X) = R then EQ = 1 else 0 If CON(X) < R then LT = 1 else 0 |
| 1000 | JUMP X | Get next instruction from memory location X |
| 1001 | JUMPGT X | Get next instruction from memory loc. X if GT=1 |
| ... | JUMPEQ X | xx = EQ |
| ... | JUMPLT X | xx = LT |
| 1101 | IN X | Input an integer value and store in X |
| 1110 | OUT X | Output, in decimal notation, content of mem. loc. X |
| 1111 | HALT | Stop program execution |

CMPUT101 Introduction to Computing

(c) Yngvi Björnsson & Vadim Bulitko

20