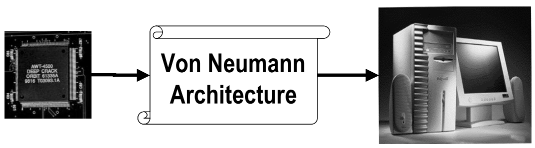
The Von Neumann Architecture
Odds and Ends

Chapter 5.1-5.2



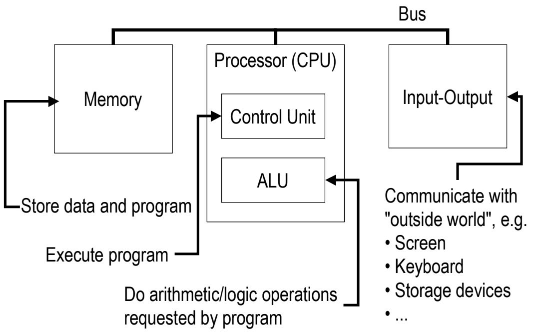
Designing Computers

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

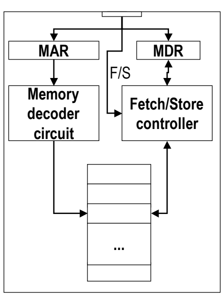
The Von Neumann Architecture

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

The Von Neumann Architecture

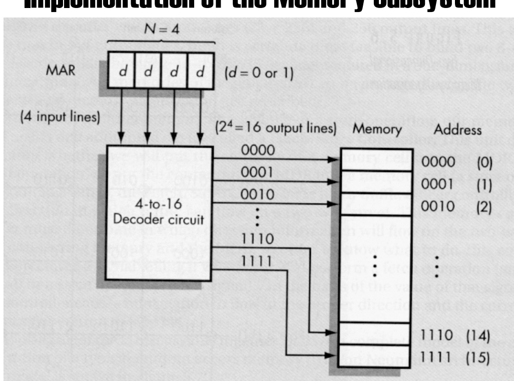


Structure of the Memory Subsystem



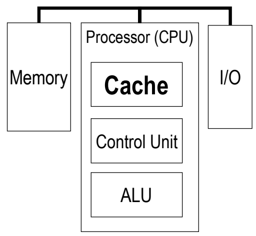
- Fetch(address)
  - Load address into MAR.
  - Decode the address in MAR.
  - Copy the content of memory cell with specified address into MDR.
- Store(address, value)
  - Load the address into MAR.
  - Load the value into MDR.
  - Decode the address in MAR
  - Copy the content of MDR into memory cell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

Implementation of the Memory Subsystem



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.

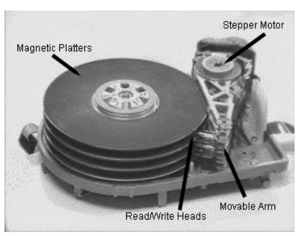


CMPUT101 Introduction to Computing (c) Yngvi Bjornsson & Vadim Bulitko 7

I/O Subsystem: Hard-Drives

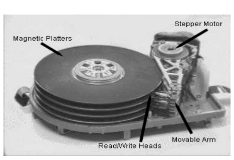
- Uses magnetic <u>surfaces</u> to store the data.
  - Each surface has many circular <u>tracks</u>.
  - Each track consists of many <u>sectors</u>.

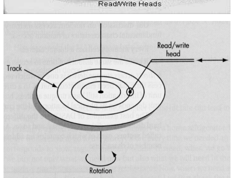
The surfaces rotate at a high speed
Typically ~7000 rev/min
The read/write arm moves:
back and forth to locate a track

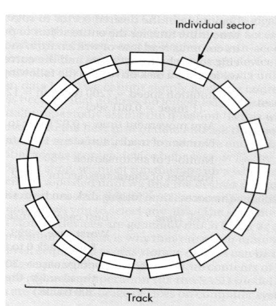


CMPUT101 Introduction to Computing (c) Yngvi Bjornsson & Vadim Bulitko 8

Hard-Drive







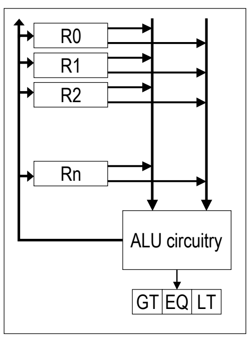
CMPUT101 Introduction to Computing (c) Yngvi Bjornsson & Vadim Bulitko 9

Disk Access Time

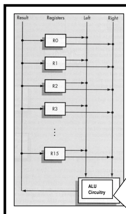
- The time it takes to read/write data to a disk, consists of:
  - Seek time
    - The time it takes to position the read/write head over correct track (depends on arm movement speed).
  - Latency
    - The time waiting for the beginning of the desired sector to get under the read/write head (depends 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)

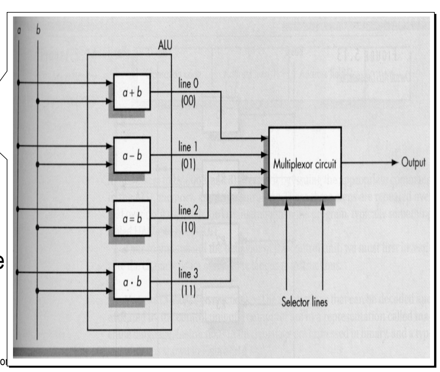
Structure of the ALU

- 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.



Implementation of the ALU

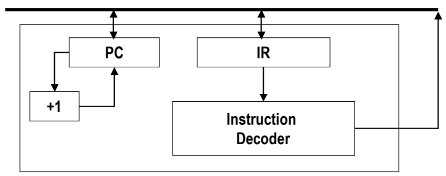




Every circuit produces a result but only the desired one is selected

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

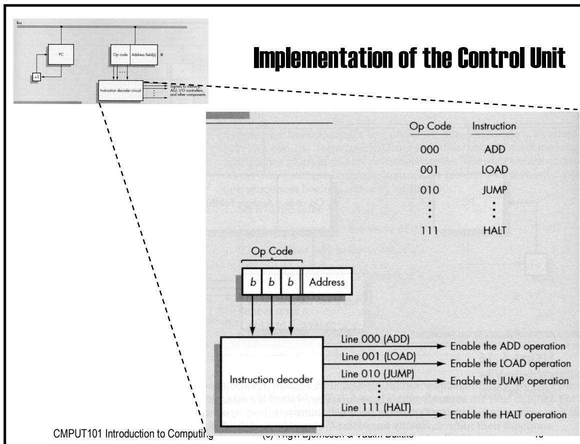


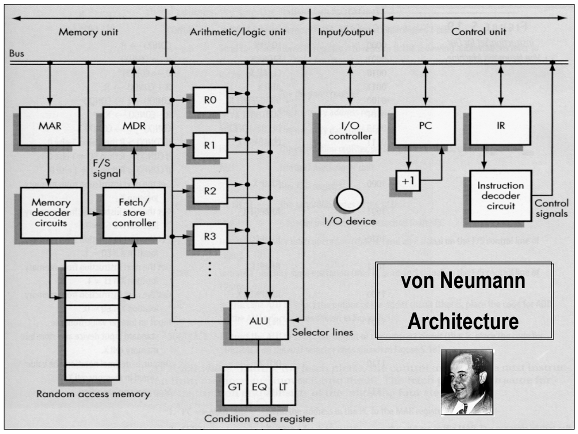
Machine Language Instructions

- 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	0000000001100011	0000000001100100

Implementation of the Control Unit





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

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

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

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
...	JUMPxx X	xx = LT / EQ / NEQ
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