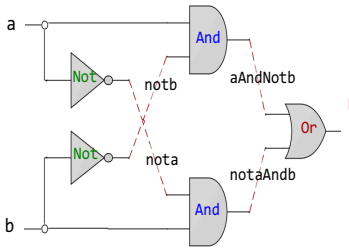
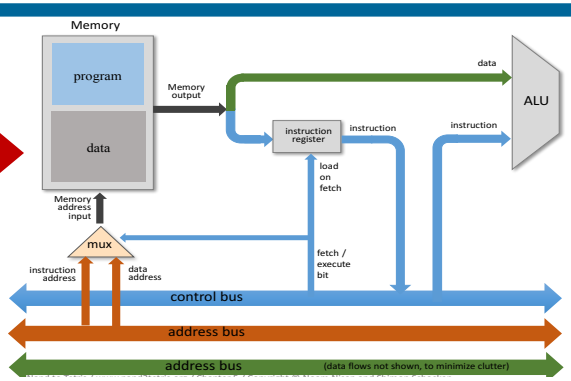
Computer Organization & Assembly Language Programming




```
CHIP Xor {
  IN a, b;
  OUT out;
  PARTS:
  Not(in=a, out=nota);
  Not(in=b, out=notb);
  And(a=nota, b=b, out=w1);
  And(a=a, b=notb, out=w2);
  Or(a=w1, b=w2, out=out);
}
```




```
@R1
D=M
@temp
M=D
```

```
0000000000000001
1111110000010000
0000000000010000
1110001100001000
```

Lecture # 16

Hack Machine Language – I

```
#include<stdio.h>
#include<stdlib.h>
int main(){
  printf("Learning is fun with Arif\n");
  exit(0);
}
```

```
global main
SECTION .data
   msg: db "Learning is fun with Arif", 0Ah, 0h
   len_msg: equ $ - msg
SECTION .text
   main:
      mov rax,1
      mov rdi,1
      mov rsi,msg
      mov rdx,len_msg
      syscall
      mov rax,60
      mov rdi,0
      syscall
```

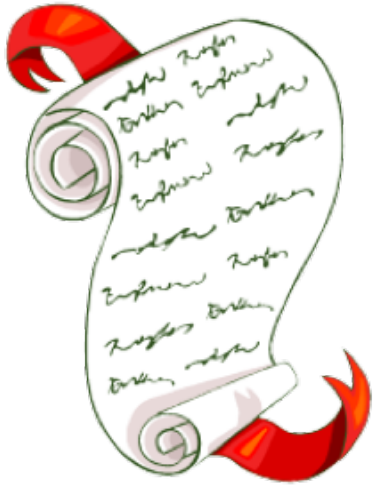
```
0:  b8 01 00 00 00
5:  bf 01 00 00 00
a:  48 be 00 00 00 00 00
11: 00 00 00
14: ba 1b 00 00 00
19: 0f 05
1b: b8 3c 00 00 00
20: bf 00 00 00 00
25: 0f 05
```

Slides of first half of the course are adapted from:
https://www.nand2tetris.org
Download s/w tools required for first half of the course from the following link:
https://drive.google.com/file/d/0B9c0BdDJz6XpZUh3X2dPR1o0MUE/view

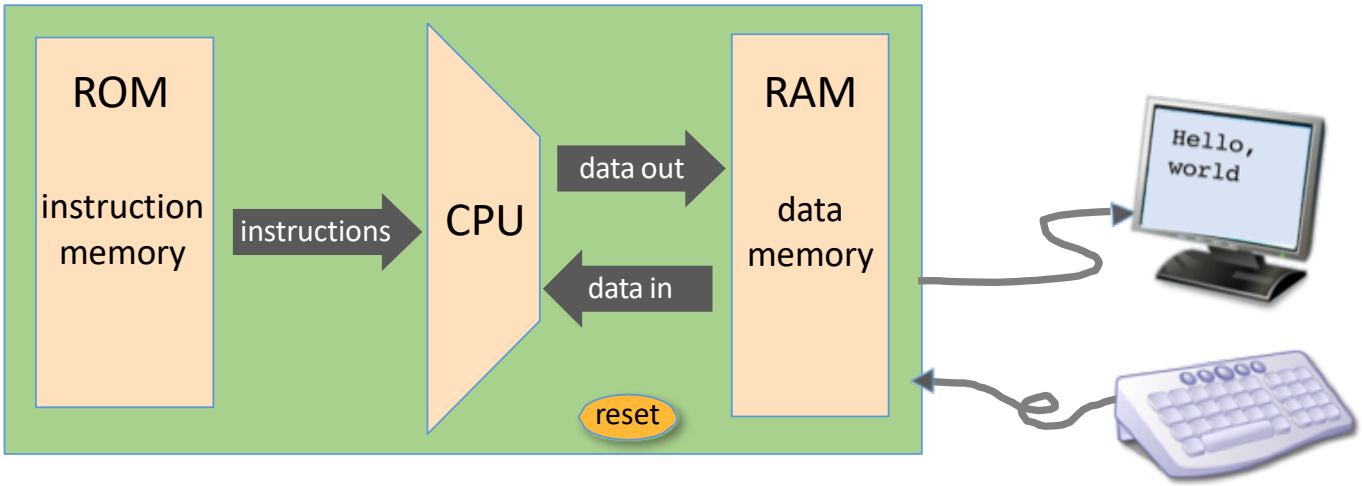
Instructor: Muhammad Arif Butt, Ph.D.

Today's Agenda

- Hack Computer Machine Language
- Review of h/w of Hack Computer
- Software of Hack Computer
  - A Instruction
  - C Instruction
  - Examples



Hack Computer: Hardware



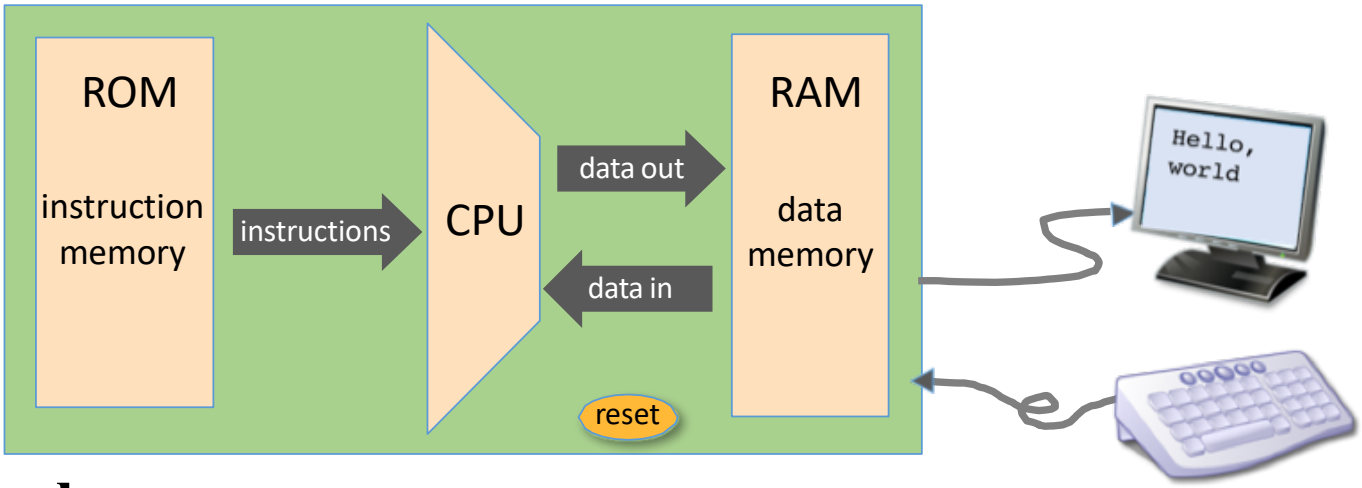
Hack computer is a 16-bit machine consisting of:

- Central Processing Unit (CPU): performs 16-bit instructions
- Data memory (RAM): a sequence of 16-bit registers having 15 bit addr:

  `RAM[0], RAM[1], RAM[2],…`
- Instruction memory (ROM): a sequence of 16-bit registers having 15 bit addr:

  `ROM[0], ROM[1], ROM[2],…`
- Two memory-mapped I/O devices: a screen and a keyboard
- Instruction bus / data bus / address buses

Hack Computer: Software



Hack machine language:

- 16-bit A-instructions
- 16-bit C-instructions

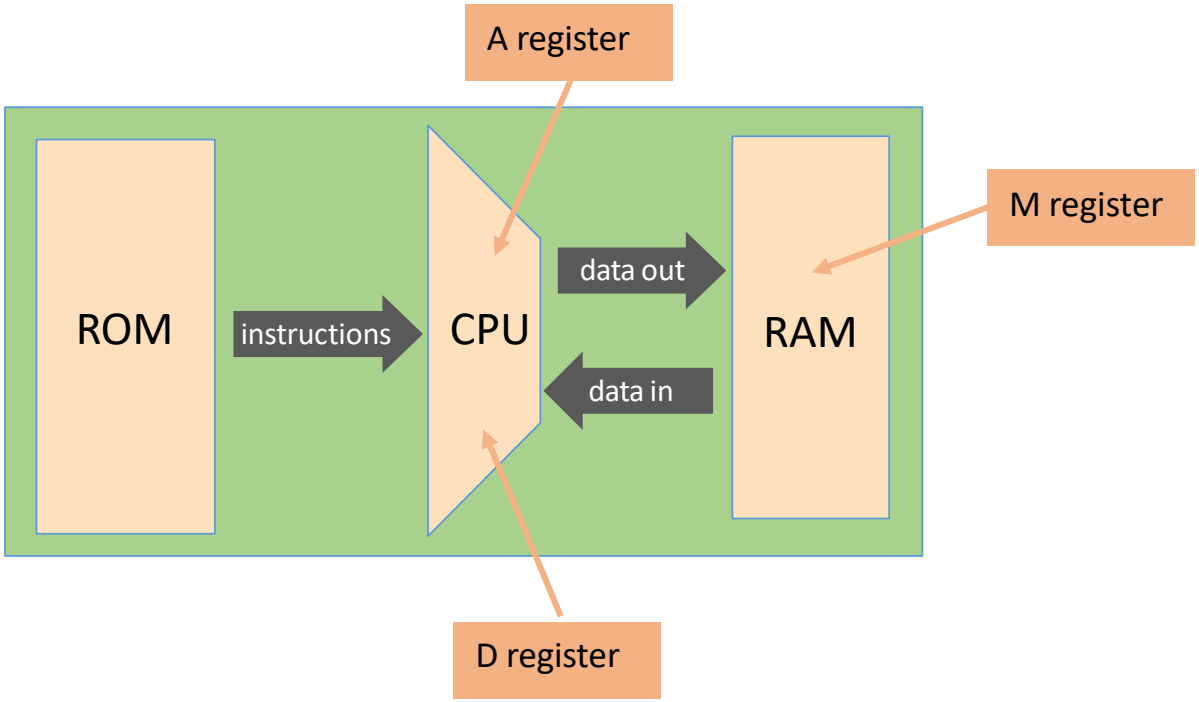
Hack program:

- A sequence of instructions written in the hack machine language

Control:

- The ROM is loaded with a Hack program (16 bit instructions)
- The reset button is pushed
- The program starts running

Hack Computer: Registers



The Hack machine language recognizes three 16-bit registers:

- `D`: used to hold data value
- `A`: used to hold data value / address of the memory
- `M`: represents the currently selected memory register: `M = RAM[A]`

The A-Instruction

The A-instruction is used to set the A register to a 15 bit value

Syntax: `@ value` Where value is either:

- A non-negative decimal constant (<= $2^{15} - 1$) or
- A symbol referring to such a constant

Semantics: Sets the A register to value, so after this

- RAM[A] becomes the selected RAM register
- ROM[A] becomes the selected ROM register

Example:

```
@17   //A =17
```

- Sets the register A to the value of 17
- As a side effect the RAM[17] becomes the selected RAM register

The C Instruction

Syntax: `dest= comp ;jump` (either dest or jump fields may be empty)

comp:

```
0, 1, -1, D, A, M, !D, !A, !M, -D, -A, -M,
D+1, A+1, M+1, D-1, A-1, M-1,
D+A, D-A, A-D, D&A, D|A,
D+M, D-M, M-D, D&M, D|M
```

dest: `null, M, D, A, MD, AM, AD, AMD`

jump: `null, JGT, JEQ, JGE, JLT, JNE, JLE, JMP`

Working:

A C instruction can be used in either of the following two ways:

- Store the result at some destination `dest= comp`
- Use the result of the computation to jump `comp ;jump`

Examples: Hack Machine Instructions

The A-instruction:

Syntax: `@value`

The C-instruction:

Syntax: `dest= comp ;jump`

comp:

```
0, 1, -1, D, A, M, !D, !A, !M, -D, -A, -M,
D+1, A+1, M+1, D-1, A-1, M-1,
D+A, D-A, A-D, D&A, D|A,
D+M, D-M, M-D, D&M, D|M
```

dest:

```
null, M, D, A, MD, AM, AD, AMD
```

jump:

```
null, JGT, JEQ, JGE, JLT, JNE, JLE, JMP
```

Example 1: Set register D to a value of -1

```
D=-1
```

```
dest= comp
```

Examples: Hack Machine Instructions

The A-instruction:

Syntax: `@value`

The C-instruction:

Syntax: `dest= comp ;jump`

comp:

```
0, 1, -1, D, A, M, !D, !A, !M, -D, -A, -M,
D+1, A+1, M+1, D-1, A-1, M-1,
D+A, D-A, A-D, D&A, D|A,
D+M, D-M, M-D, D&M, D|M
```

dest: `null, M, D, A, MD, AM, AD, AMD`

jump: `null, JGT, JEQ, JGE, JLT, JNE, JLE, JMP`

Example 2: Suppose the programmer wants to increment the value of D

```
D=D+1
```

```
dest= comp
```

Examples: Hack Machine Instructions

The A-instruction:

Syntax: `@value`

The C-instruction:

Syntax: `dest= comp ;jump`

comp:

```
0, 1, -1, D, A, M, !D, !A, !M, -D, -A, -M,
D+1, A+1, M+1, D-1, A-1, M-1,
D+A, D-A, A-D, D&A, D|A,
D+M, D-M, M-D, D&M, D|M
```

dest: `null, M, D, A, MD, AM, AD, AMD`

jump: `null, JGT, JEQ, JGE, JLT, JNE, JLE, JMP`

Example 3: Suppose the programmer wants to add the contents of D and A-register and place the result in D-register

```
D=D+A
```

```
dest= comp
```

Examples: Hack Machine Instructions

The A-instruction:

Syntax: `@value`

The C-instruction:

Syntax: `dest= comp ;jump`

comp:

```
0, 1, -1, D, A, M, !D, !A, !M, -D, -A, -M,
D+1, A+1, M+1, D-1, A-1, M-1,
D+A, D-A, A-D, D&A, D|A,
D+M, D-M, M-D, D&M, D|M
```

dest: `null, M, D, A, MD, AM, AD, AMD`

jump: `null, JGT, JEQ, JGE, JLT, JNE, JLE, JMP`

Example 4: Suppose the programmer wants to store a number 10 in register D

```
@10   //A =10
D=A
```

```
@value
dest= comp
```

Examples: Hack Machine Instructions

The A-instruction:

Syntax: `@value`

The C-instruction:

Syntax: `dest= comp ;jump`

comp:

```
0, 1, -1, D, A, M, !D, !A, !M, -D, -A, -M,
D+1, A+1, M+1, D-1, A-1, M-1,
D+A, D-A, A-D, D&A, D|A,
D+M, D-M, M-D, D&M, D|M
```

dest: `null, M, D, A, MD, AM, AD, AMD`

jump: `null, JGT, JEQ, JGE, JLT, JNE, JLE, JMP`

Example 5: Suppose the programmer wants to write the value of register D at RAM[135]

```
@135   //A =135
M=D
```

```
@value
dest= comp
```

Examples: Hack Machine Instructions

The A-instruction:

Syntax: `@value`

The C-instruction:

Syntax: `dest= comp ;jump`

comp:

```
0, 1, -1, D, A, M, !D, !A, !M, -D, -A, -M,
D+1, A+1, M+1, D-1, A-1, M-1,
D+A, D-A, A-D, D&A, D|A,
D+M, D-M, M-D, D&M, D|M
```

dest:

```
null, M, D, A, MD, AM, AD, AMD
```

jump:

```
null, JGT, JEQ, JGE, JLT, JNE, JLE, JMP
```

Example 6: Suppose the programmer wants to write the value of register D+1 at RAM[135]

```
@135   //A =135
M=D+1
```

```
@value
dest= comp
```

Examples: Hack Machine Instructions

The A-instruction:

Syntax: `@value`

The C-instruction:

Syntax: `dest= comp ;jump`

comp:

```
0, 1, -1, D, A, M, !D, !A, !M, -D, -A, -M,
D+1, A+1, M+1, D-1, A-1, M-1,
D+A, D-A, A-D, D&A, D|A,
D+M, D-M, M-D, D&M, D|M
```

dest:

```
null, M, D, A, MD, AM, AD, AMD
```

jump:

```
null, JGT, JEQ, JGE, JLT, JNE, JLE, JMP
```

Example 7: Suppose the programmer wants to read memory contents from address 325 and place them in D register

```
@325   //A =325
D=M //D=M[325]
```

```
@value
dest= comp
```

Examples: Hack Machine Instructions

The A-instruction:

Syntax: `@value`

The C-instruction:

Syntax: `dest= comp ;jump`

comp:
```
0, 1, -1, D, A, M, !D, !A, !M, -D, -A, -M,
D+1, A+1, M+1, D-1, A-1, M-1,
D+A, D-A, A-D, D&A, D|A,
D+M, D-M, M-D, D&M, D|M
```

dest:
```
null, M, D, A, MD, AM, AD, AMD
```

jump:
```
null, JGT, JEQ, JGE, JLT, JNE, JLE, JMP
```

Example 8: Suppose the programmer wants to do an unconditional jump to ROM[431]

```
@431 //A =431
0;JMP
```

```
@value
comp; jump
```

Examples: Hack Machine Instructions

The A-instruction:

Syntax: `@value`

The C-instruction:

Syntax: `dest= comp ;jump`

comp:

```
0, 1, -1, D, A, M, !D, !A, !M, -D, -A, -M,
D+1, A+1, M+1, D-1, A-1, M-1,
D+A, D-A, A-D, D&A, D|A,
D+M, D-M, M-D, D&M, D|M
```

dest:

```
null, M, D, A, MD, AM, AD, AMD
```

jump:

```
null, JGT, JEQ, JGE, JLT, JNE, JLE, JMP
```

Example 9: Suppose the programmer wants to jump to ROM[97], if D-1 == 0

```
@97 //A =97
D-1;JEQ
```

```
@value
comp; jump
```

Examples: Hack Machine Instructions

The A-instruction:

Syntax: `@value`

The C-instruction:

Syntax: `dest= comp ;jump`

comp:

```
0, 1, -1, D, A, M, !D, !A, !M, -D, -A, -M,
D+1, A+1, M+1, D-1, A-1, M-1,
D+A, D-A, A-D, D&A, D|A,
D+M, D-M, M-D, D&M, D|M
```

dest:

```
null, M, D, A, MD, AM, AD, AMD
```

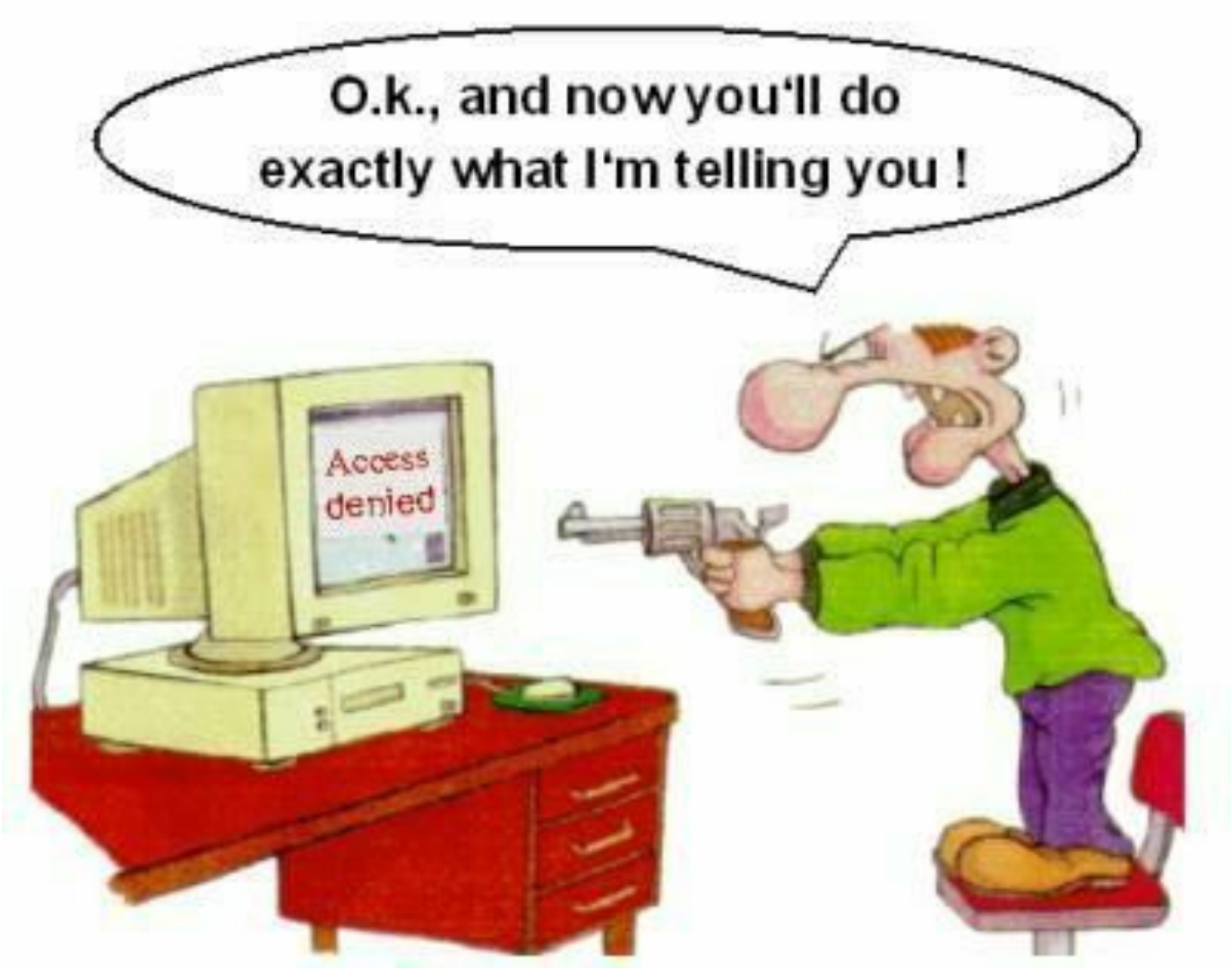
jump:

```
null, JGT, JEQ, JGE, JLT, JNE, JLE, JMP
```

Example 10: Suppose the programmer wants to write constant 54 at RAM[17]

```
//D=54
@54
D=A
//M[17]=D
@17
M=D
```

Things To Do


O.k., and now you'll do exactly what I'm telling you !
Access denied