การจัดองค์การคอมพิวเตอร์

w4.6 ภาษาแอสเซมบลี (1/3)

31110321 Computer Organization
สำหรับนักศึกษาชั้นปีที่ 3 สาขาวิชาวิศวกรรมคอมพิวเตอร์

ทรงฤทธิ์ กิติศรีวรพันธุ์

songrit@npu.ac.th

สาขาวิชาวิศวกรรมคอมพิวเตอร์
มหาวิทยาลัยนครพนม

Revised 2020-08-16

Lecture plan

- 4.1 ภาษาเครื่อง
- 4.2 ส่วนประกอบพื้นฐาน
- 4.3 ระบบแฮกค์คอมพิวเตอร์และภาษาเครื่อง
- 4.4 ภาษาเครื่องแฮกค์
- 4.5 อินพุท / เอาท์พุท
- **4.6 การเขียนโปรแกรมสำหรับเครื่องแฮกค์ (1-3)**
- 4.7 ภาพรวมโปรเจ็คสัปดาห์4

Hack assembly language (overview)

A-instruction:

```
@value      // A = value
```

C-instruction:

```
dest = comp ; jump
```

(both *dest* and *jump* are optional)

where:

comp =

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

dest =

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

(M refers to RAM[A])

jump =

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

- ประมวลผล **comp**
- เก็บข้อมูลลง **dest**
- ถ้า **comp jump 0** เป็นจริง
  - Jump ไปทำคำสั่งในหน่วยความจำ ROM[A]

Hack assembler

Assembly program

```
// Program: Flip.asm
// flips the values of
// RAM[0] and RAM[1]
   @R1
   D=M
   @temp
   M=D    // temp = R1

   @R0
   D=M
   @R1
   M=D    // R1 = R0

   @temp
   D=M
   @R0
   M=D    // R0 = temp

(END)
   @END
   0;JMP
```

Hack assembler

Binary code

```
0000000000000001
1111110000010000
0000000000010000
1110001100001000
0000000000000000
1111110000010000
0000000000000001
1110001100001000
0000000000010000
1111110000010000
0000000000000000
1110001100001000
0000000000001100
1110101010000111
```

execute

CPU Emulator

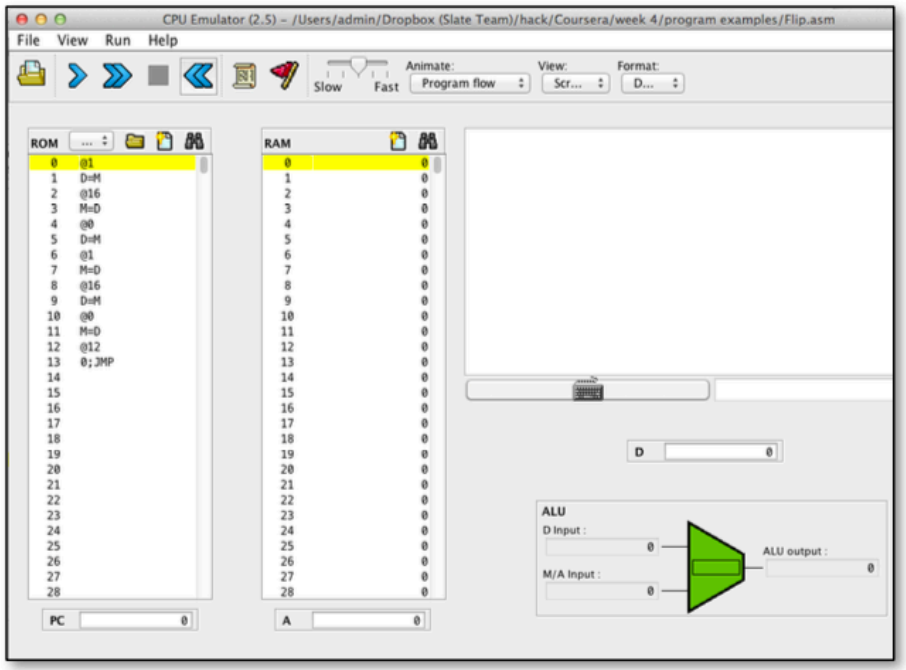
Assembly program

```
// Program: Flip.asm
// flips the values of
// RAM[0] and RAM[1]
   @R1
   D=M
   @temp
   M=D     // temp = R1

   @R0
   D=M
   @R1
   M=D     // R1 = R0

   @temp
   D=M
   @R0
   M=D     // R0 = temp

(END)
   @END
   0;JMP
```

load

แปลงนีโมนิก
เป็นรหัสภาษาเครื่อง

CPU Emulator



Hack programming

- การติดต่อรีจีสเตอร์และหน่วยความจำ } Unit 4.6
- Branching
- Variables
- Iteration

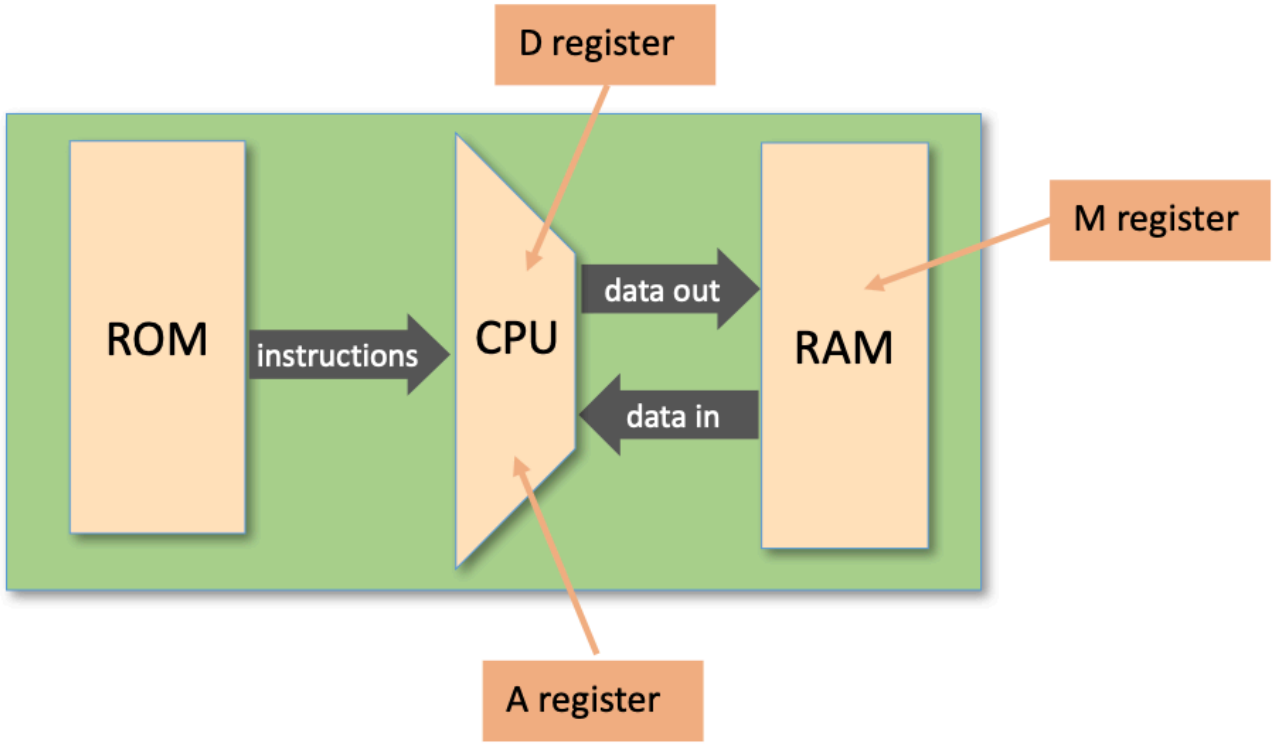
} Unit 4.7

- Pointers
- Input/output

} Unit 4.8

Registers and memory

- **D** : data register
- **A** : address/ data register
- **M** : ตำแหน่งหน่วยความจำที่ถูกเลือก M=RAM[A]



Registers and memory

- D : data register
- A : address/ data register
- M : ตำแหน่งหน่วยความจำที่ถูกเลือก M=RAM[A]

Typical operations:

```
// D=10
```

ตัวอย่างโปรแกรม: บวกเลข 2 จำนวน

Hack assembly code

```
// Program: Add2.asm
// Computes: RAM[2] = RAM[0] + RAM[1]
// Usage: put values in RAM[0], RAM[1]
```

translate and load

(white space ignored)

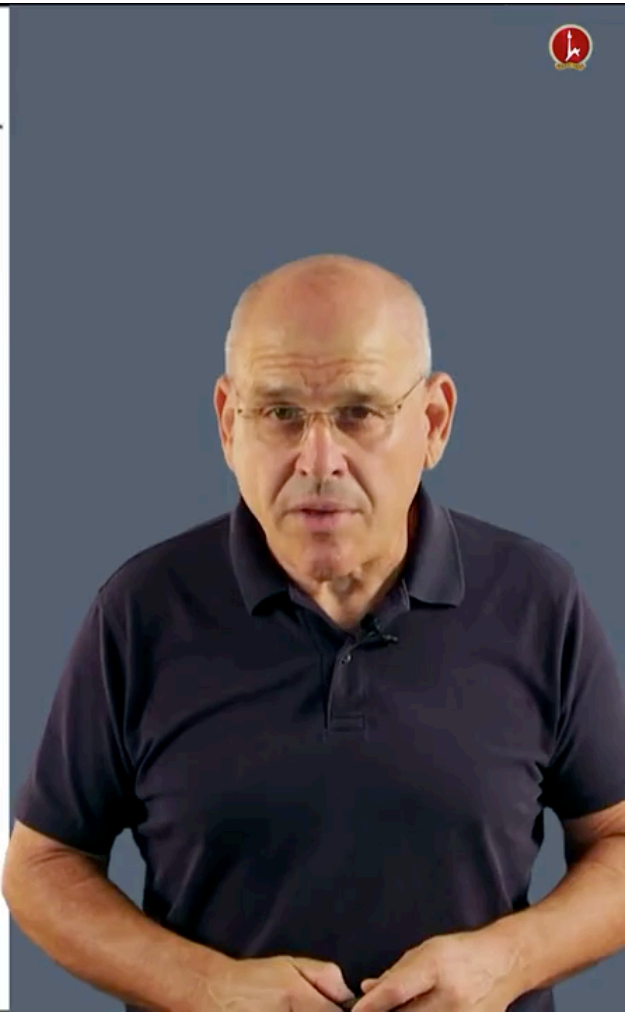
Memory (ROM)

0	0000000000000000
1	1111110000010000
2	0000000000000001
3	1111000010010000
4	0000000000000010
5	1110001100001000
6	
7	
8	
9	
10	
11	
12	
13	
14	
15	
	⋮
32767	

binary view

CPU Emulator

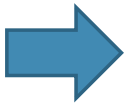
Hack program example: add two numbers



ออกจากโปรแกรม

Hack assembly code

```
   // Program: Add2.asm
   // Computes: RAM[2] = RAM[0] + RAM[1]
   // Usage: put values in RAM[0], RAM[1]

0  @0
1  D=M    // D = RAM[0]

2  @1
3  D=D+M  // D = D + RAM[1]

4  @2
5  M=D    // RAM[2] = D
```



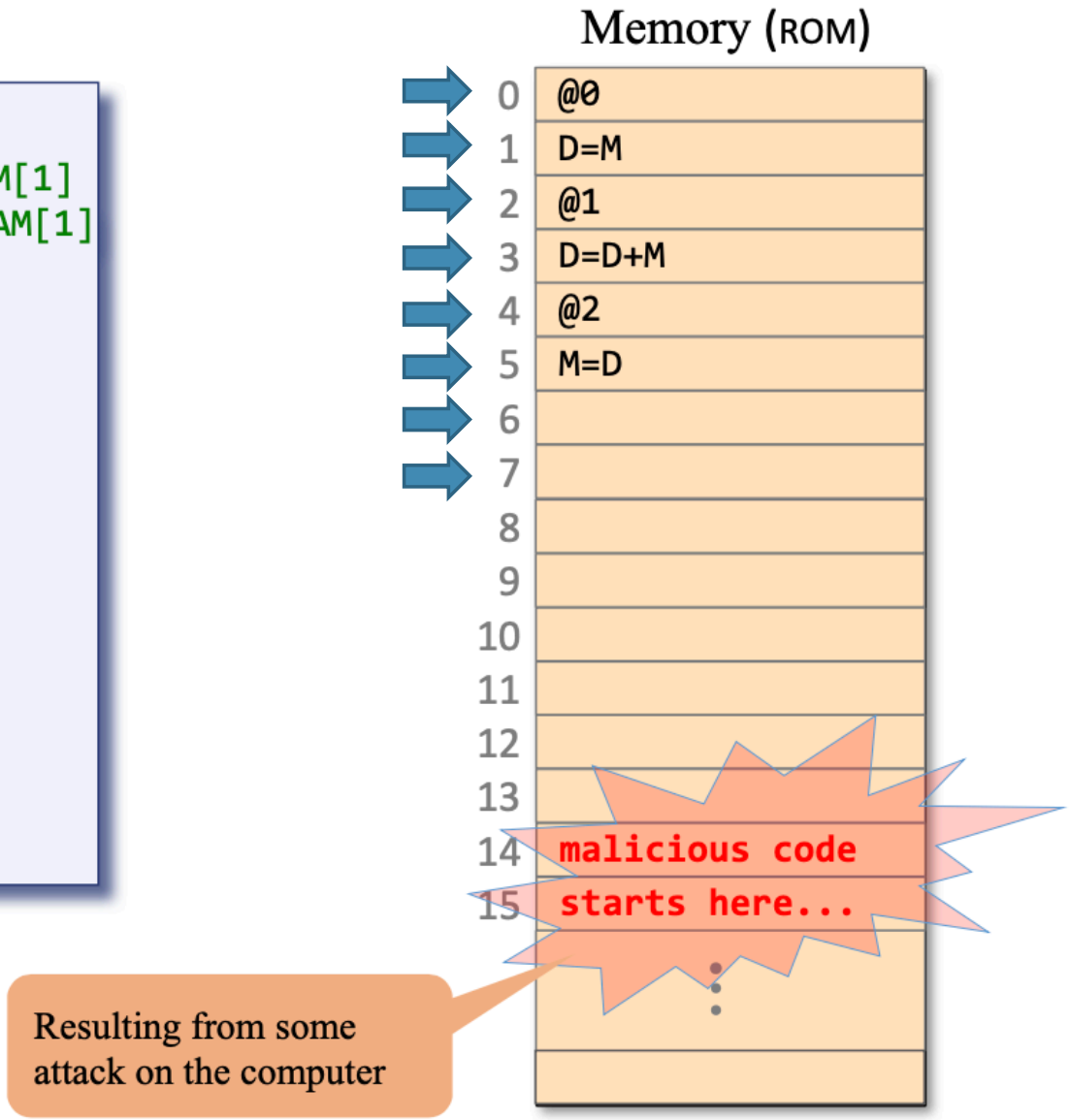
Memory (ROM)

0	@0
1	D=M
2	@1
3	D=D+M
4	@2
5	M=D
6	
7	
8	
9	
10	
11	
12	
13	
14	
15	
	⋮

ออกโปรแกรม

Hack assembly code

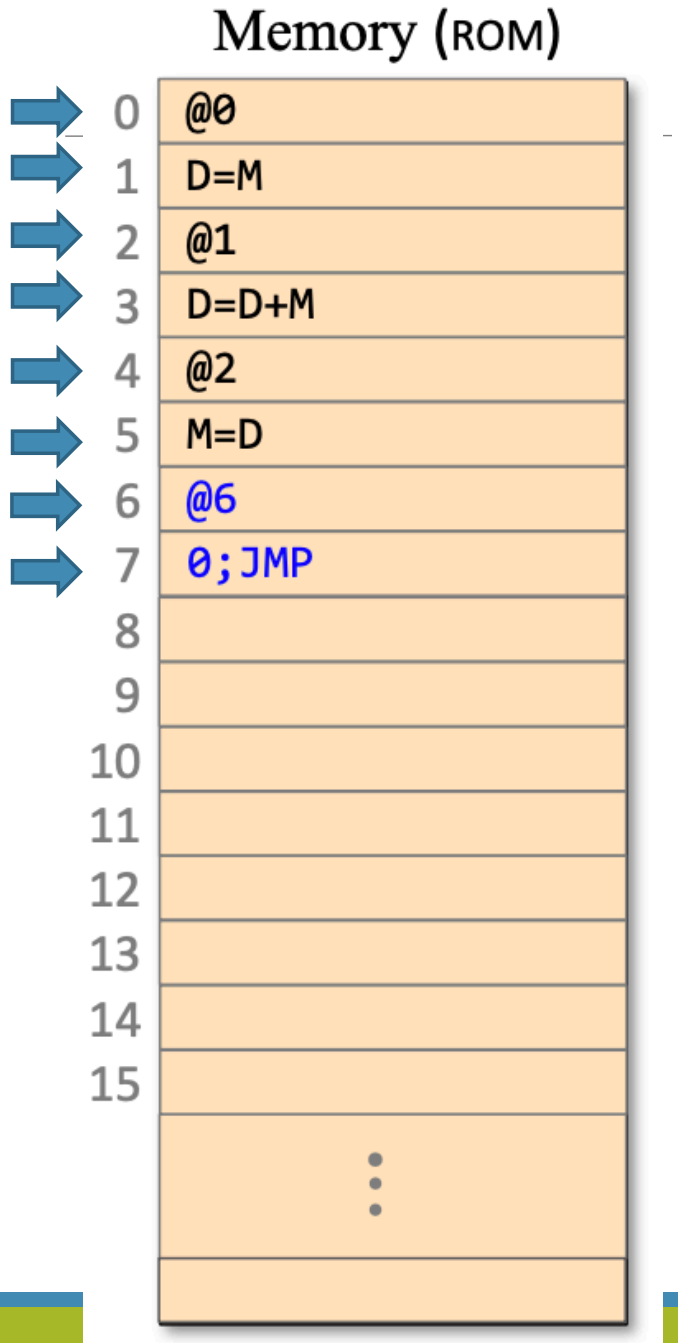
```
   // Program: Add2.asm
   // Computes: RAM[2] = RAM[0] + RAM[1]
   // Usage: put values in RAM[0], RAM[1]

0  @0
1  D=M    // D = RAM[0]

2  @1
3  D=D+M  // D = D + RAM[1]

4  @2
5  M=D    // RAM[2] = D
```



ออกโปรแกรม

Hack assembly code

```
   // Program: Add2.asm
   // Computes: RAM[2] = RAM[0] + RAM[1]
   // Usage: put values in RAM[0], RAM[1]

0  @0
1  D=M    // D = RAM[0]

2  @1
3  D=D+M  // D = D + RAM[1]

4  @2
5  M=D    // RAM[2] = D

6
7
```



Best practice:

To terminate a program safely, end it with an infinite loop.

Built-in symbols

- มีการจับคู่สัญลักษณ์ภายในระบบมาก่อน

symbol	value
R0	0
R1	1
R2	2
...	...
R15	15

Attention: Hack is case-sensitive!
`R5` and `r5` are different symbols.

- ตัวอย่างต้องการเซต RAM[5] = 7

implementation:

```
// let RAM[5] = 7
@7
D=A

@5
M=D
```

better style:

```
// let RAM[5] = 7
@7
D=A

@R5
M=D
```

Built-in symbols

symbol	value
R0	0
R1	1
R2	2
...	...
R15	15
SCREEN	16384
KBD	24576

symbol	value
SP	0
LCL	1
ARG	2
THIS	3
THAT	4

- R0, R1, ..., R15 แทนรีจีสเตอร์(เสมือน) ใช้แทนตัวแปรได้
- SCREEN และ KBD ใช้แทนตำแหน่งจอภาพ และ คีย์บอร์ด
- และข้อมูลสัญลักษณ์อื่น อ่านได้ที่ ch7-8

Lecture plan

- 4.1 ภาษาเครื่อง
- 4.2 ส่วนประกอบพื้นฐาน
- 4.3 ระบบแฮกค์คอมพิวเตอร์และภาษาเครื่อง
- 4.4 ภาษาเครื่องแฮกค์
- 4.5 อินพุท / เอาท์พุท
- **4.6 การเขียนโปรแกรมสำหรับเครื่องแฮกค์ (2/3)**
- 4.7 ภาพรวมโปรเจ็ค