การจัดองค์การคอมพิวเตอร์

w4.4 ภาษาเครื่องแฮกค์

31110321 Computer Organization
สำหรับนักศึกษาชั้นปีที่ 3 สาขาวิชาวิศวกรรมคอมพิวเตอร์

ทรงฤทธิ์ กิติศรีวรพันธุ์

songrit@npu.ac.th

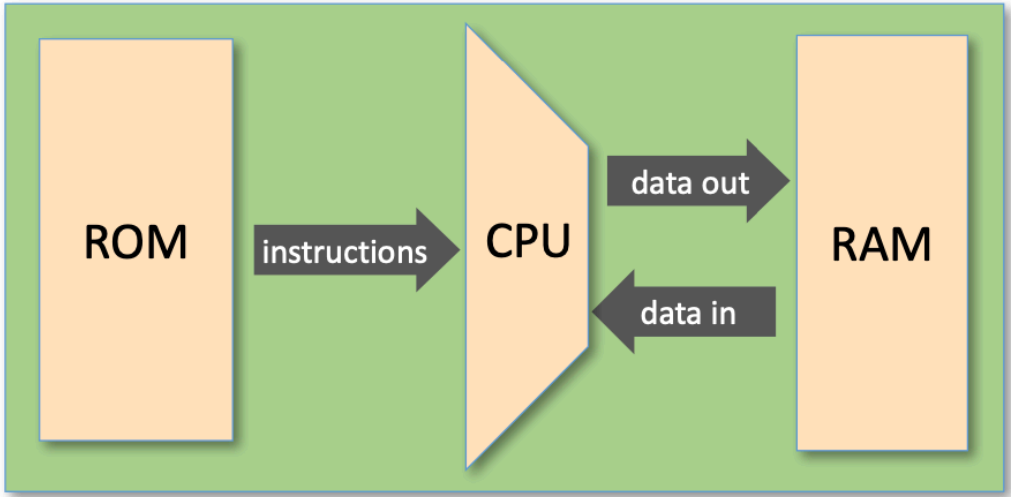
สาขาวิชาวิศวกรรมคอมพิวเตอร์
มหาวิทยาลัยนครพนม

Revised 2020-08-16

Lecture plan

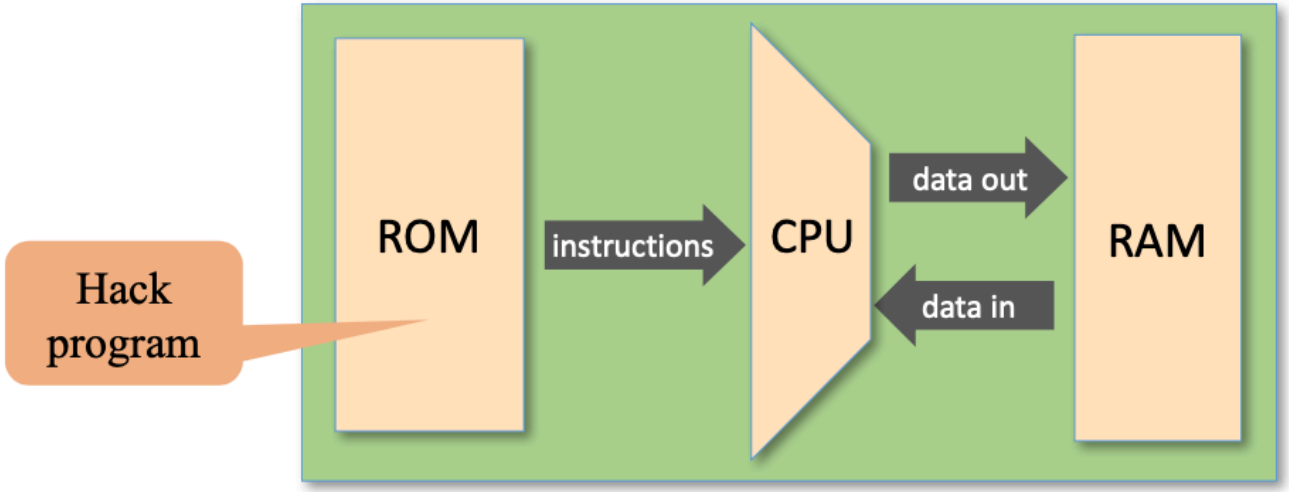
- 4.1 ภาษาเครื่อง
- 4.2 ส่วนประกอบพื้นฐาน
- 4.3 ระบบแฮกค์คอมพิวเตอร์และภาษาเครื่อง
- **4.4 ภาษาเครื่องแฮกค์**
- 4.5 อินพุท / เอาท์พุท
- 4.6 การเขียนโปรแกรมสำหรับเครื่องแฮกค์
- 4.7 ภาพรวมโปรเจ็คสัปดาห์4

ระบบแฮกค์คอมพิวเตอร์ : ซอฟต์แวร์



- ภาษาเครื่อง
  - A-instruction ขนาด 16-bit
  - C-instruction ขนาด 16-bit
- Hack program คือชุดคำสั่งภาษาเครื่องเขียนเป็นลำดับ

Hack machine language



Symbolic:

```
@17
D+1;JLE
```

Mnemonic

Binary:

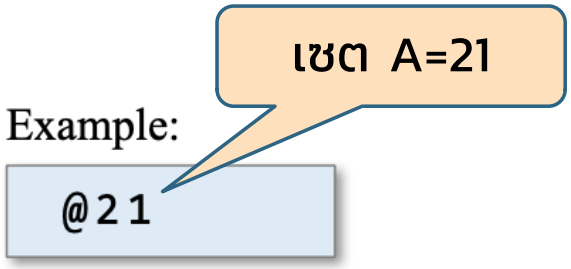
```
0000000000010001
1110011111000110
```

Binary code

A-instruction

- หน้าที่ : เซตค่าให้รีจีสเตอร์ A
- Symbolic syntax:

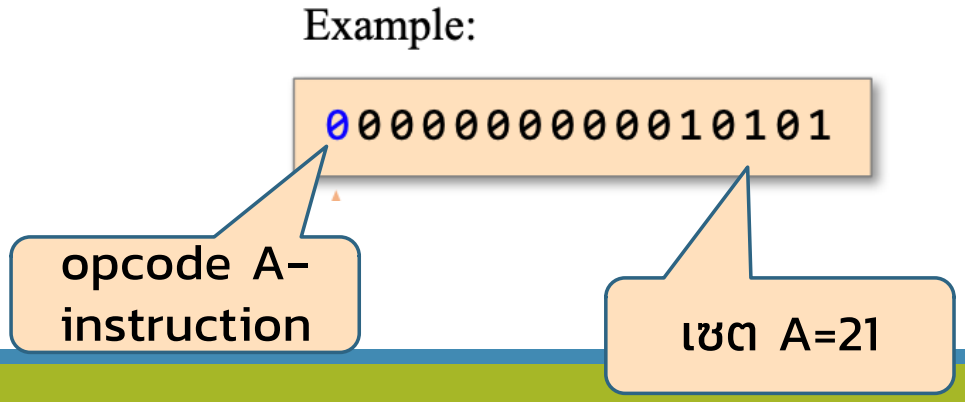
@value



  - เมื่อ value คือ
    - เลขจำนวนเต็มบวกตั้งแต่ 0 ≤ 65535 (2^{15}-1)
    - หรือ ชื่อ (label)
- Binary syntax:

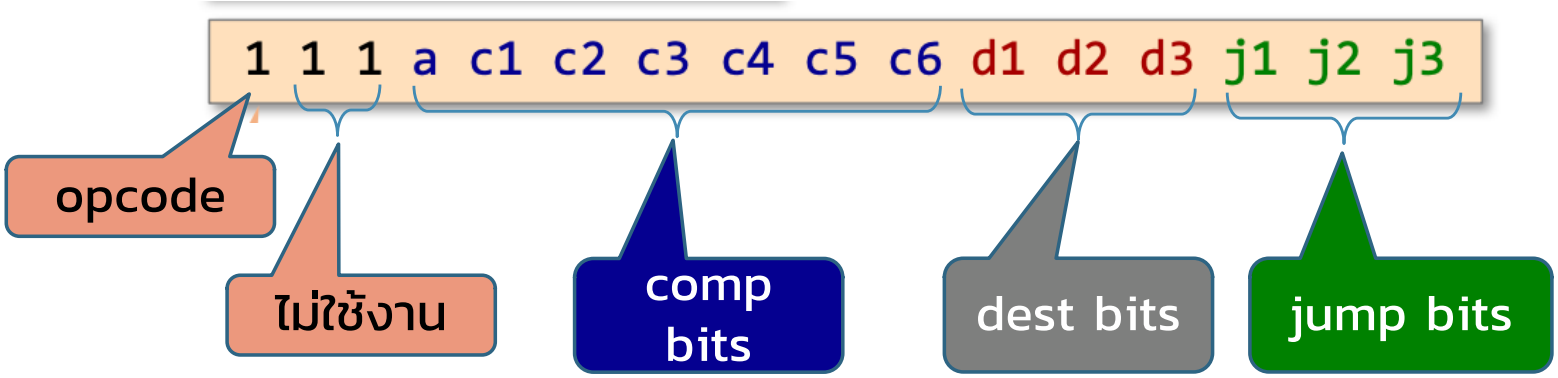
0value

value คือรหัสไบนารี 15-bit



C-instruction

- หน้าที่ : Flow control
- Symbolic syntax: `dest = comp ; jump`
- Binary syntax:



C-instruction : comp

- หน้าที่ : Flow control
- Symbolic syntax:

dest = *comp* ; *jump*

- Binary syntax:

comp		c1	c2	c3	c4	c5	c6
0		1	0	1	0	1	0
1		1	1	1	1	1	1
-1		1	1	1	0	1	0
D		0	0	1	1	0	0
A	M	1	1	0	0	0	0
!D		0	0	1	1	0	1
!A	!M	1	1	0	0	0	1
-D		0	0	1	1	1	1
-A	-M	1	1	0	0	1	1
D+1		0	1	1	1	1	1
A+1	M+1	1	1	0	1	1	1
D-1		0	0	1	1	1	0
A-1	M-1	1	1	0	0	1	0
D+A	D+M	0	0	0	0	1	0
D-A	D-M	0	1	0	0	1	1
A-D	M-D	0	0	0	1	1	1
D&A	D&M	0	0	0	0	0	0
D|A	D|M	0	1	0	1	0	1
a==0	a==1						

C-instruction : dest

- หน้าที่ : Flow control
- Symbolic syntax: *dest* = *comp* ; *jump*
- Binary syntax: `1 1 1 a c1 c2 c3 c4 c5 c6 d1 d2 d3 j1 j2 j3`

dest	d1 d2 d3	effect: the value is stored in:
null	0 0 0	The value is not stored
M	0 0 1	RAM[A]
D	0 1 0	D register
MD	0 1 1	RAM[A] and D register
A	1 0 0	A register
AM	1 0 1	A register and RAM[A]
AD	1 1 0	A register and D register
AMD	1 1 1	A register, RAM[A], and D register

C-instruction : jump

- หน้าที่ : Flow control
- Symbolic syntax: *dest* = *comp* ; *jump*
- Binary syntax: `1 1 1 a c1 c2 c3 c4 c5 c6 d1 d2 d3 j1 j2 j3`

jump	j1	j2	J3	effect
null	0	0	0	no jump
JGT	0	0	1	if out>0 jump
JEQ	0	1	0	if out=0 jump
JGE	0	1	1	if out≥0 jump
JLT	1	0	0	if out<0 jump
JNE	1	0	1	if out≠0 jump
JLE	1	1	0	if out≤0 jump
JMP	1	1	1	unconditional jump

C-instruction : MD=D+1

Symbolic syntax: *dest* = *comp* ; *jump*

Binary syntax: `1 1 1 a c1 c2 c3 c4 c5 c6 d1 d2 d3 j1 j2 j3`

comp		c1	c2	c3	c4	c5	c6
0		1	0	1	0	1	0
1		1	1	1	1	1	1
-1		1	1	1	0	1	0
D		0	0	1	1	0	0
A	M	1	1	0	0	0	0
!D		0	0	1	1	0	1
!A	!M	1	1	0	0	0	1
-D		0	0	1	1	1	1
-A	-M	1	1	0	0	1	1
D+1		0	1	1	1	1	1
A+1	M+1	1	1	0	1	1	1
D-1		0	0	1	1	1	0
A-1	M-1	1	1	0	0	1	0
D+A	D+M	0	0	0	0	1	0
D-A	D-M	0	1	0	0	1	1
A-D	M-D	0	0	0	1	1	1
D&A	D&M	0	0	0	0	0	0
D|A	D|M	0	1	0	1	0	1
a==0	a==1						

dest	d1	d2	d3	effect: the value is stored in:
null	0	0	0	The value is not stored
M	0	0	1	RAM[A]
D	0	1	0	D register
MD	0	1	1	RAM[A] and D register
A	1	0	0	A register
AM	1	0	1	A register and RAM[A]
AD	1	1	0	A register and D register
AMD	1	1	1	A register, RAM[A], and D register

jump	j1	j2	j3	effect:
null	0	0	0	no jump
JGT	0	0	1	if out > 0 jump
JEQ	0	1	0	if out = 0 jump
JGE	0	1	1	if out ≥ 0 jump
JLT	1	0	0	if out < 0 jump
JNE	1	0	1	if out ≠ 0 jump
JLE	1	1	0	if out ≤ 0 jump
JMP	1	1	1	Unconditional jump

Examples:

Symbolic: `MD=D+1`

Binary: `1110011111011000`

C-instruction : M=1

Symbolic syntax: *dest* = *comp* ; *jump*

Binary syntax: `1 1 1 a c1 c2 c3 c4 c5 c6 d1 d2 d3 j1 j2 j3`

comp		c1	c2	c3	c4	c5	c6
0		1	0	1	0	1	0
1		1	1	1	1	1	1
-1		1	1	1	0	1	0
D		0	0	1	1	0	0
A	M	1	1	0	0	0	0
!D		0	0	1	1	0	1
!A	!M	1	1	0	0	0	1
-D		0	0	1	1	1	1
-A	-M	1	1	0	0	1	1
D+1		0	1	1	1	1	1
A+1	M+1	1	1	0	1	1	1
D-1		0	0	1	1	1	0
A-1	M-1	1	1	0	0	1	0
D+A	D+M	0	0	0	0	1	0
D-A	D-M	0	1	0	0	1	1
A-D	M-D	0	0	0	1	1	1
D&A	D&M	0	0	0	0	0	0
D|A	D|M	0	1	0	1	0	1
a==0	a==1						

dest	d1	d2	d3	effect: the value is stored in:
null	0	0	0	The value is not stored
M	0	0	1	RAM[A]
D	0	1	0	D register
MD	0	1	1	RAM[A] and D register
A	1	0	0	A register
AM	1	0	1	A register and RAM[A]
AD	1	1	0	A register and D register
AMD	1	1	1	A register, RAM[A], and D register

jump	j1	j2	j3	effect:
null	0	0	0	no jump
JGT	0	0	1	if out > 0 jump
JEQ	0	1	0	if out = 0 jump
JGE	0	1	1	if out ≥ 0 jump
JLT	1	0	0	if out < 0 jump
JNE	1	0	1	if out ≠ 0 jump
JLE	1	1	0	if out ≤ 0 jump
JMP	1	1	1	Unconditional jump

Examples:

Symbolic:

```
M=1
```

Binary:

```
1110111111001000
```

C-instruction : M=1

Symbolic syntax: *dest* = *comp* ; *jump*

Binary syntax: `1 1 1 a c1 c2 c3 c4 c5 c6 d1 d2 d3 j1 j2 j3`

comp		c1	c2	c3	c4	c5	c6
0		1	0	1	0	1	0
1		1	1	1	1	1	1
-1		1	1	1	0	1	0
D		0	0	1	1	0	0
A	M	1	1	0	0	0	0
!D		0	0	1	1	0	1
!A	!M	1	1	0	0	0	1
-D		0	0	1	1	1	1
-A	-M	1	1	0	0	1	1
D+1		0	1	1	1	1	1
A+1	M+1	1	1	0	1	1	1
D-1		0	0	1	1	1	0
A-1	M-1	1	1	0	0	1	0
D+A	D+M	0	0	0	0	1	0
D-A	D-M	0	1	0	0	1	1
A-D	M-D	0	0	0	1	1	1
D&A	D&M	0	0	0	0	0	0
D|A	D|M	0	1	0	1	0	1
a==0	a==1						

dest	d1	d2	d3	effect: the value is stored in:
null	0	0	0	The value is not stored
M	0	0	1	RAM[A]
D	0	1	0	D register
MD	0	1	1	RAM[A] and D register
A	1	0	0	A register
AM	1	0	1	A register and RAM[A]
AD	1	1	0	A register and D register
AMD	1	1	1	A register, RAM[A], and D register

jump	j1	j2	j3	effect:
null	0	0	0	no jump
JGT	0	0	1	if out > 0 jump
JEQ	0	1	0	if out = 0 jump
JGE	0	1	1	if out ≥ 0 jump
JLT	1	0	0	if out < 0 jump
JNE	1	0	1	if out ≠ 0 jump
JLE	1	1	0	if out ≤ 0 jump
JMP	1	1	1	Unconditional jump

Examples:

Symbolic: `D+1;JLE`

Binary: `1110011111000110`

Hack program

```
// Usage: put a number in RAM[0]
@16 // RAM[16] represents i
M=1 //i=1
@17 // RAM[17] represents sum
M=0 //sum=0

@16
D=M
@0
D=D-M
@17 // if i>RAM[0] goto 17
D;JGT

@16
D=M
@17
M=D+M // sum += i
@16
M=M+1 // i++
@4 // goto 4 (loop)
0;JMP

@17
D=M
@1
M=D // RAM[1] = sum
@21 // program's end
0;JMP // infinite loop
```

- สิ่งที่เรารู้จากโค้ด
  - โปรแกรมมีการทำงานเรียงตามลำดับ
  - ใช้ช่องว่างได้
  - ใช้เครื่องหมาย // แทน comment
  - มีชุดคำสั่ง
    - A-instruction
    - C-instruction

Hack program

```
// Usage: put a number in RAM[0]
@16 // RAM[16] represents i
M=1 //i=1
@17 // RAM[17] represents sum
M=0 //sum=0

@16
D=M
@0
D=D-M
@17 // if i>RAM[0] goto 17
D;JGT

@16
D=M
@17
M=D+M // sum += i
@16
M=M+1 // i++
@4 // goto 4 (loop)
0;JMP

@17
D=M
@1
M=D // RAM[1] = sum
@21 // program's end
0;JMP // infinite loop
```

translate

```
0000000000010000
1110111111001000
0000000000010001
1110101010001000
0000000000010000
1111110000010000
0000000000000000
1111010011010000
0000000000010001
1110001100000001
0000000000010000
1111110000010000
0000000000010001
1110000010001000
0000000000010000
1111110111001000
0000000000000100
1110101010000111
0000000000010001
1111110000010000
0000000000000001
1110001100001000
0000000000010101
1110101010000111
```

execute

Lecture plan

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