การจัดองค์การคอมพิวเตอร์

CPU.hdl

31110321 Computer Organization
สำหรับนักศึกษาชั้นปีที่ 3 สาขาวิชาวิศวกรรมคอมพิวเตอร์

ทรงฤทธิ์ กิติศรีวรพันธุ์

songrit@npu.ac.th

สาขาวิชาวิศวกรรมคอมพิวเตอร์
มหาวิทยาลัยนครพนม

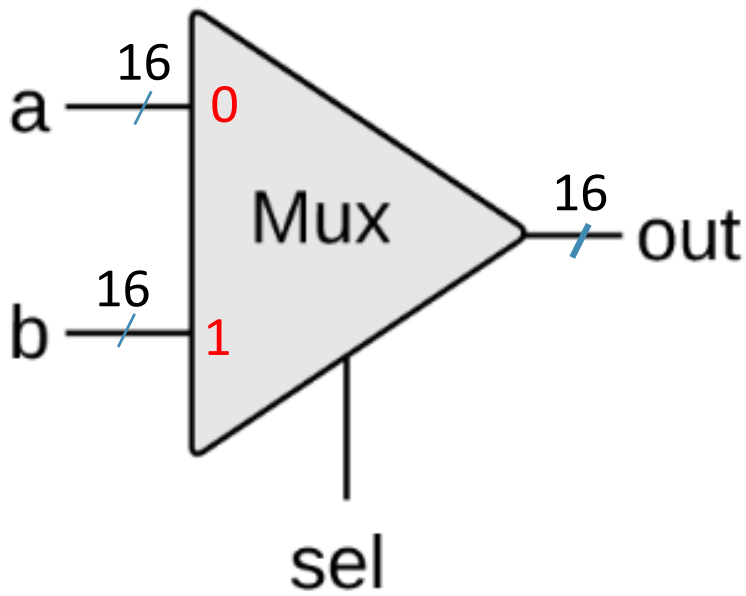
Revised 2020-10-14

Lecture plan

- 5.1 สถาปัตยกรรมฟอนนอยมันน์
- 5.2 Fetch-Execute Cycle
- 5.3 ซีพียูแฮกค์
  - **5.3-1 CPU.hdl**
- 5.4 แฮกค์คอมพิวเตอร์
- 5.5 ภาพรวมโปรเจ็ค 5
  - Computer.hdl

Mux16

- Multiplexor

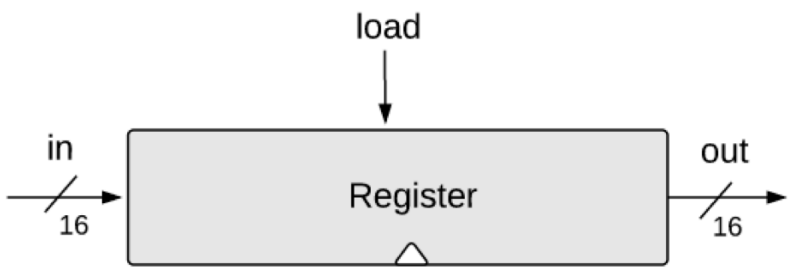


sel	out
0	a
1	b

```
if (sel==0)
    out=a
else
    out=b
```

- sel == 0 → ส่ง a ออก out
- sel == 1 → ส่ง b ออก out

ARegister และ DRegister



- in 16 บิต
- Out 16 บิต
- 1 control bit (load)

Register.hdl

```
/**
 * 16-bit register:
 * If load(t) then out(t+1) = in(t)
 * else            out(t+1) = out(t))
 */

CHIP Register {
    IN in[16], load;
    OUT out[16];

    PARTS:
    // Put your code here:
}
```

Hack CPU Implementation

- มี Instruction 2 แบบ
  - A-instruction
  - C-instruction
- ถอดรหัสโดยใช้ b_{15} ของ instruction code
  - If b_{15} == 0 then A-instruction
  - Else if b_{15} == 1 → C-instruction

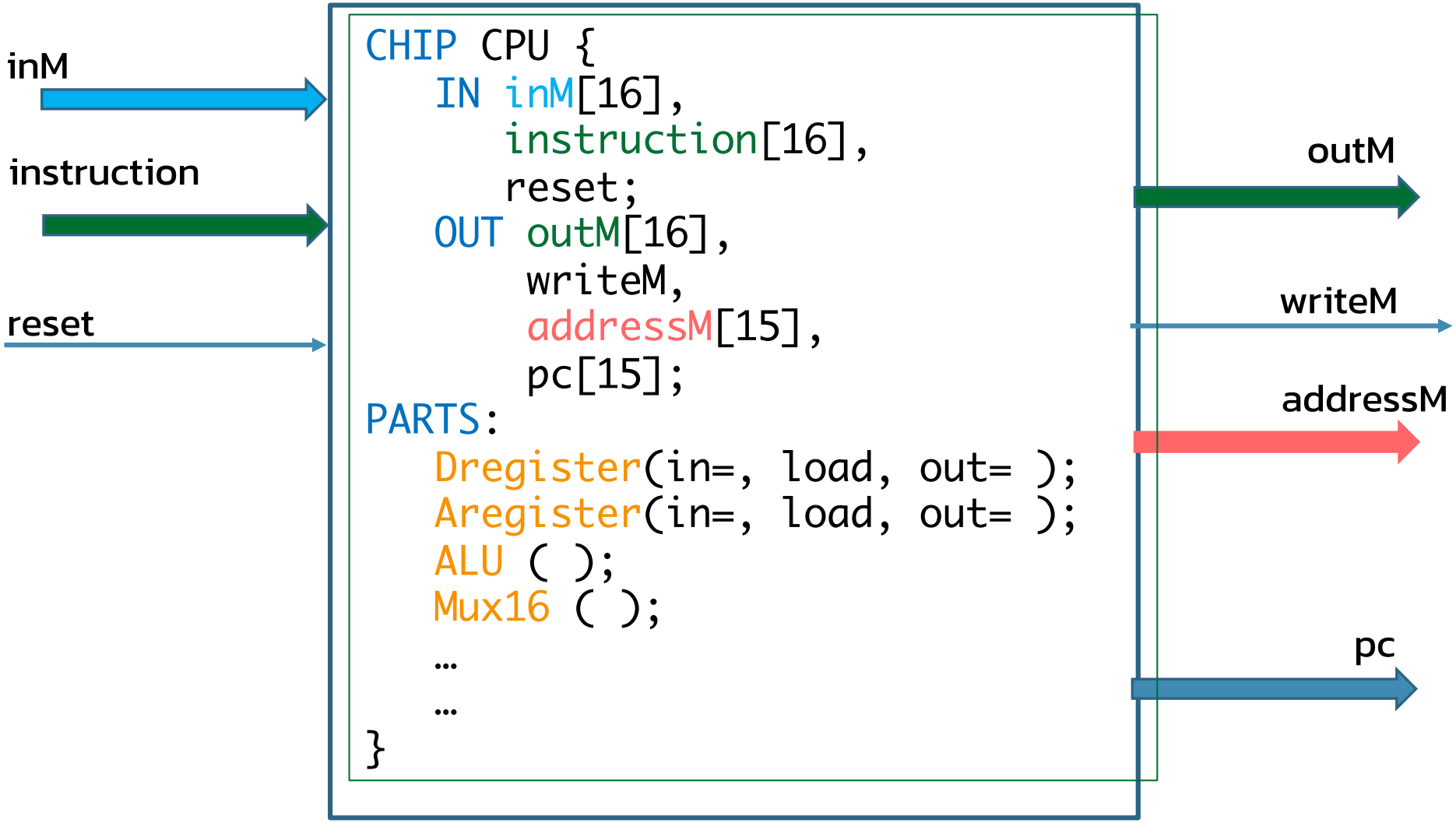
Instruction

- instruction มี 16บิท แบ่งเป็น 2 ชนิด
- บิท 15 ใช้ระบุชนิดของ instruction

A-instruction **0000 0000 0001 0101**

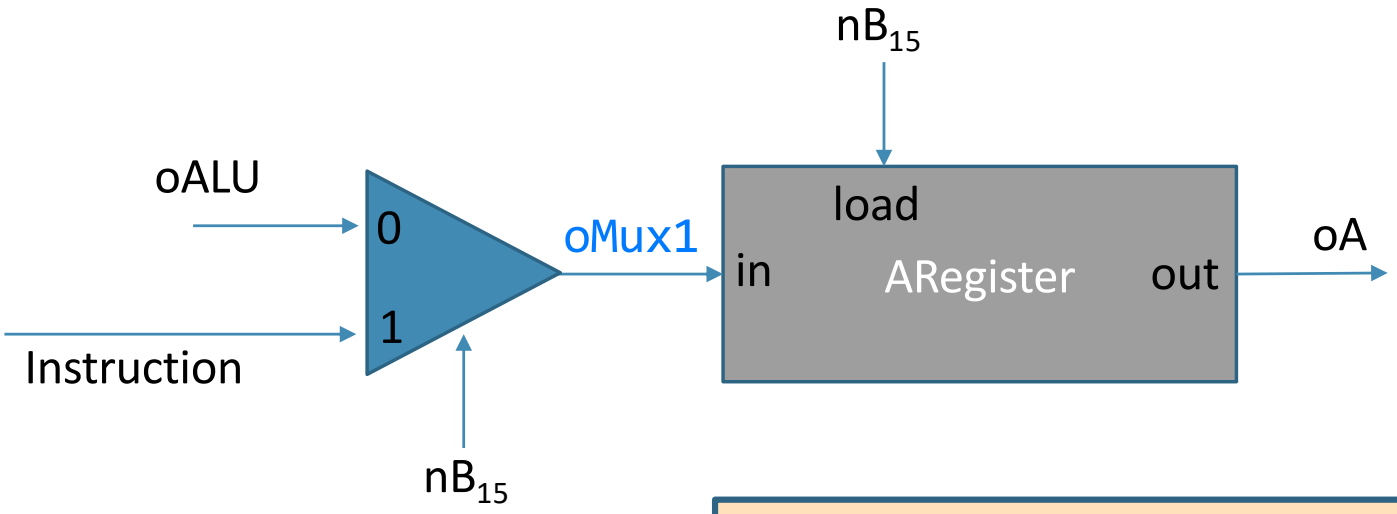
C-instruction **111A** $C_1C_2C_3C_4$ C_5C_6 D_1D_2 $D_3J_1J_2J_3$

CPU interface



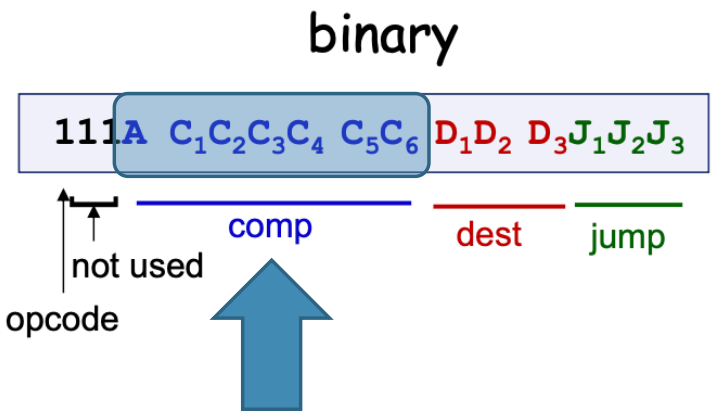
ARegister ใช้บันทึกผลคำนวณ

- ARegsiter ใช้บันทึกผลลัพธ์จาก ALU ด้วย




```
Not(in=instruction[15],
    out=nB15);
Mux16(a=oALU, b=instruction,
      sel=muxA, out=oMux1)
ARegister(in=oMux1,
          load=nB15,
           out=oA);
```

comp



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						

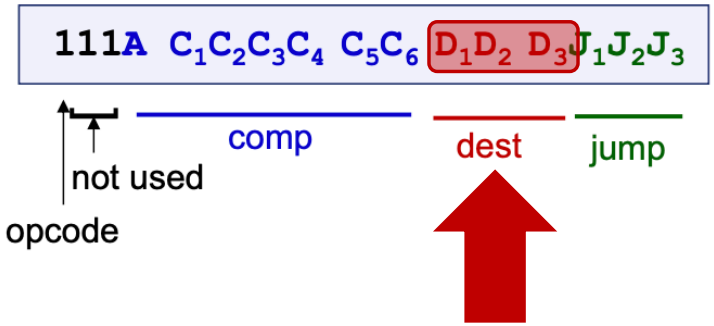
ALU

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						

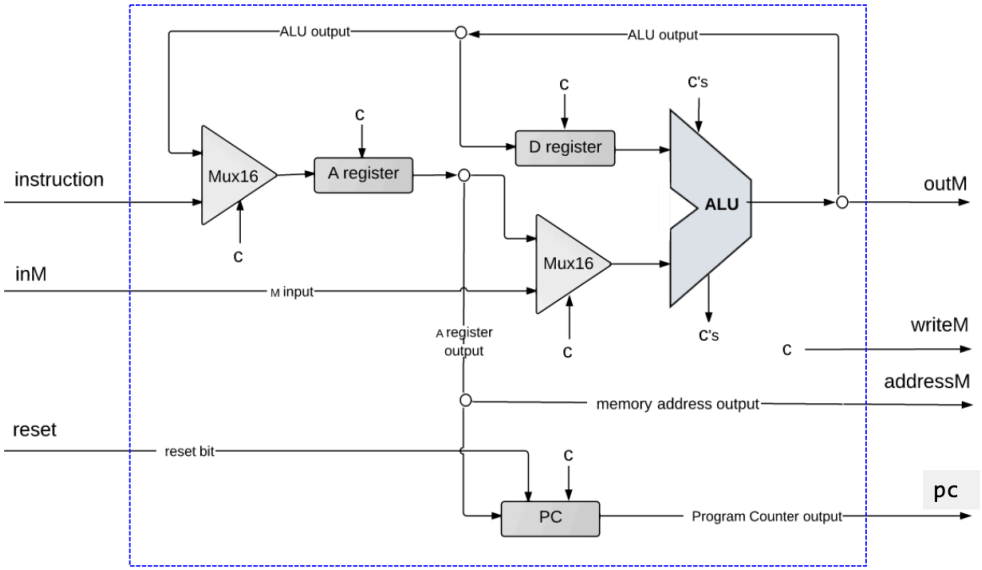
```
ALU(
//Inputs
x=oD,
y=oMux2,
zx=instruction[11], //c1
nx=instruction[10], //c2
zy=instruction[9], //c3
ny=instruction[8], //c4
f=instruction[7], //c5
no=instruction[6], //c6
//Output
out=oALU, zr=zero, ng=neg);
```

dest

binary

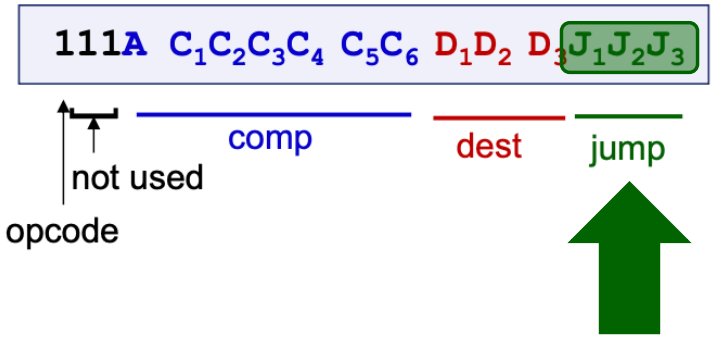


dest	d1 (b_5)	d2 (b_4)	d3 (b_3)	effect: the value is stored in:
null	0	0	0	The value is not stored
M	0	0	1	RAM[A]
D	0	1	0	D register
MD	0	1	1	RAM[A] and D register
A	1	0	0	A register
AM	1	0	1	A register and RAM[A]
AD	1	1	0	A register and D register
AMD	1	1	1	A register, RAM[A], and D register



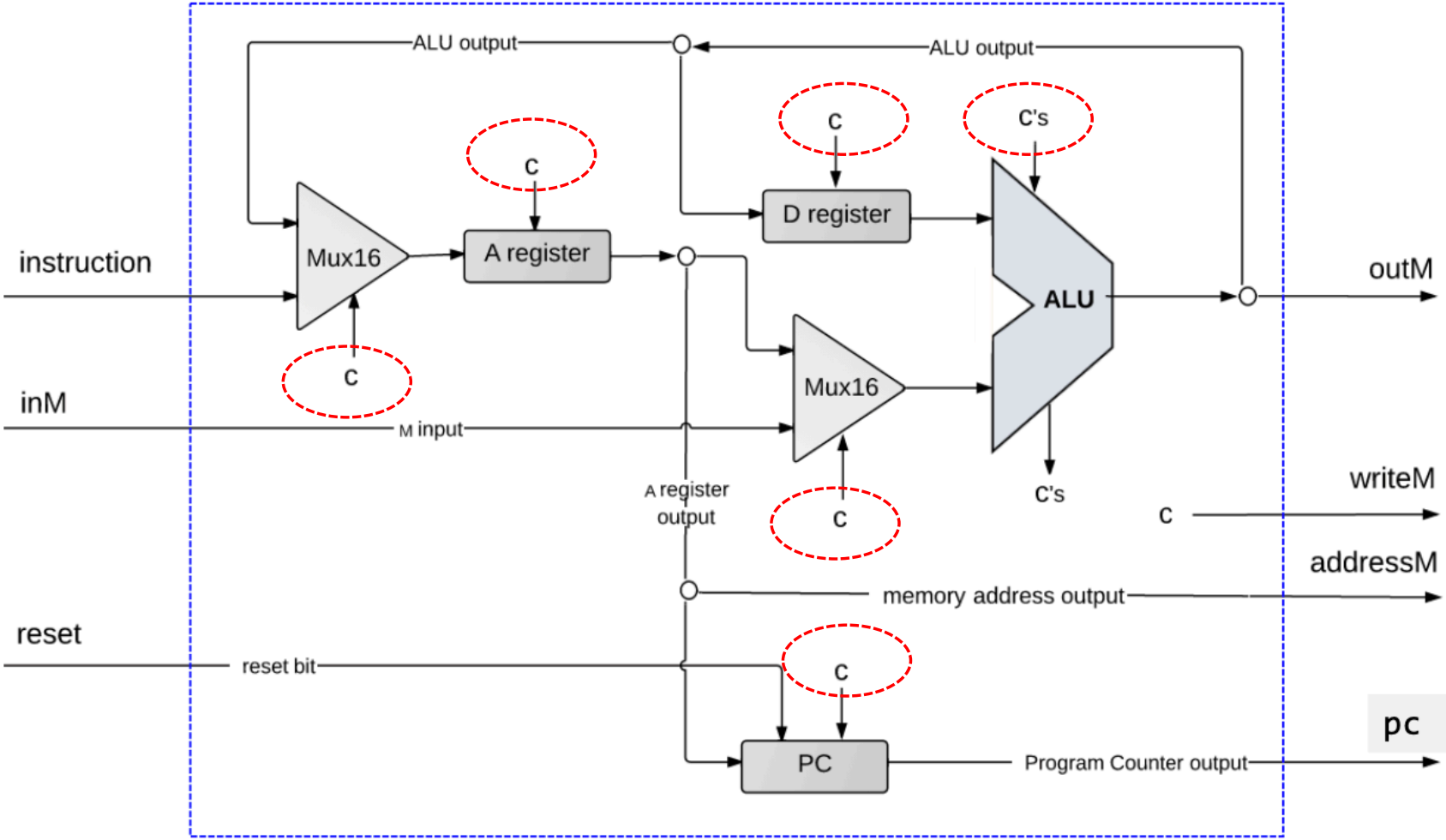
jmp

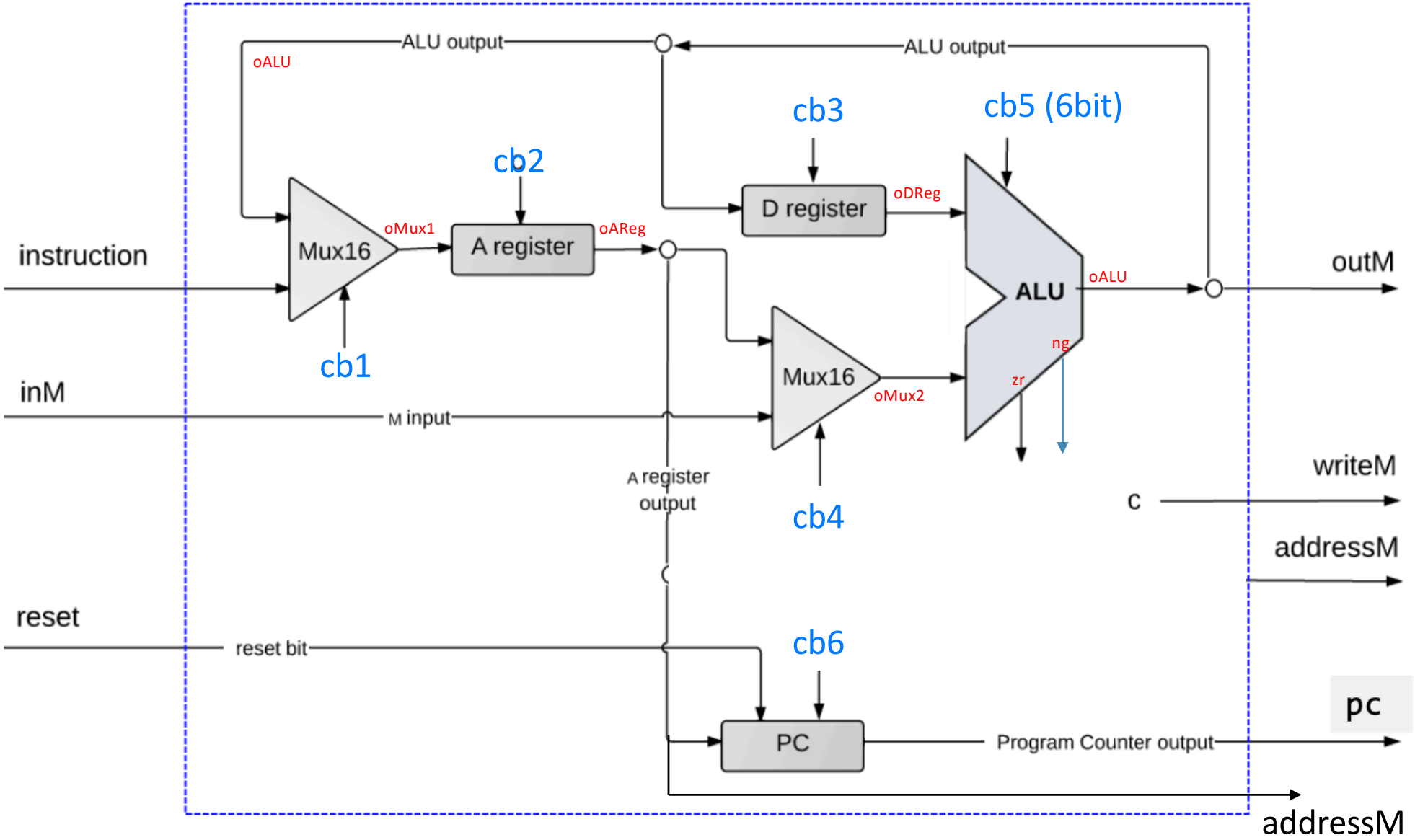
binary



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

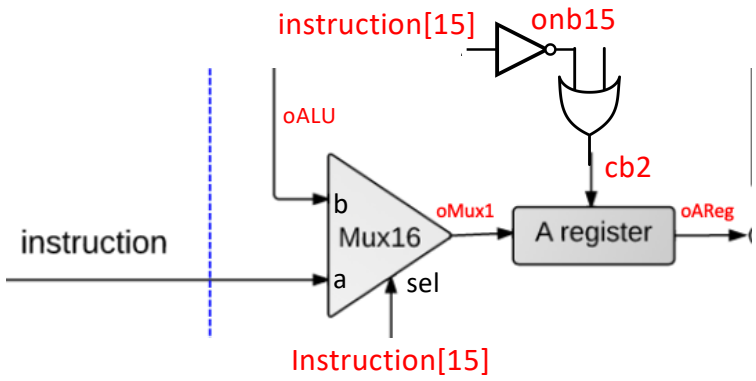
CPU : Control bit




ALU output
oALU
ALU output
cb3
cb5 (6bit)
cb2
D register
oDReg
instruction
Mux16
oMux1
A register
oAReg
ALU
oALU
outM
cb1
ng
inM
Mux16
zr
oMux2
M input
writeM
A register
output
cb4
c
addressM
reset
reset bit
cb6
pc
PC
Program Counter output
addressM

Control bit : cb1, cb2

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

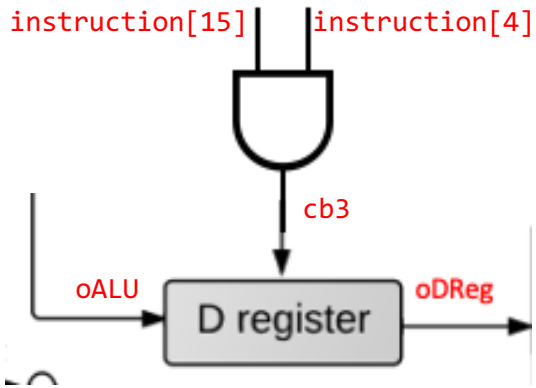



```
Mux16(a=instruction, b=oALU, sel=instruction[15], out=oMux1);
Not(in=instruction[15], out=onb15);
Or(a=onb15, b=instruction[5], out=cb2);
ARegister(in=oMux1, load=cb2, out=oAReg);
```

Control bit : cb3

b_4

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



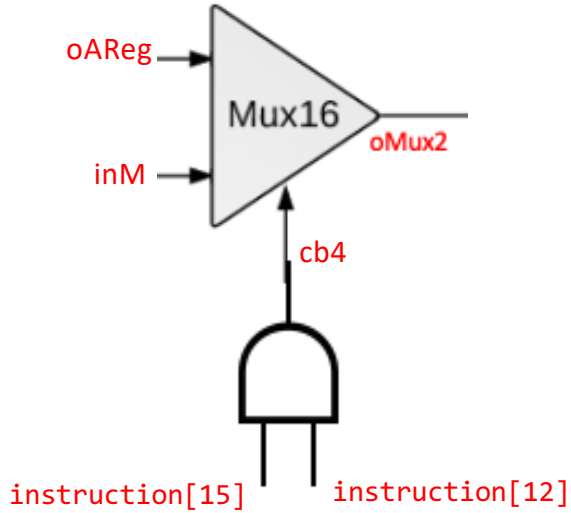

```
And(a=instruction[15], b=instruction[4], out=cb3);
DRegister(in=oALU, load=cb3, out=oDReg);
```

Control bit : cb4

- เลือกหน่วยความจำ หรือ จาก ARegister

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						

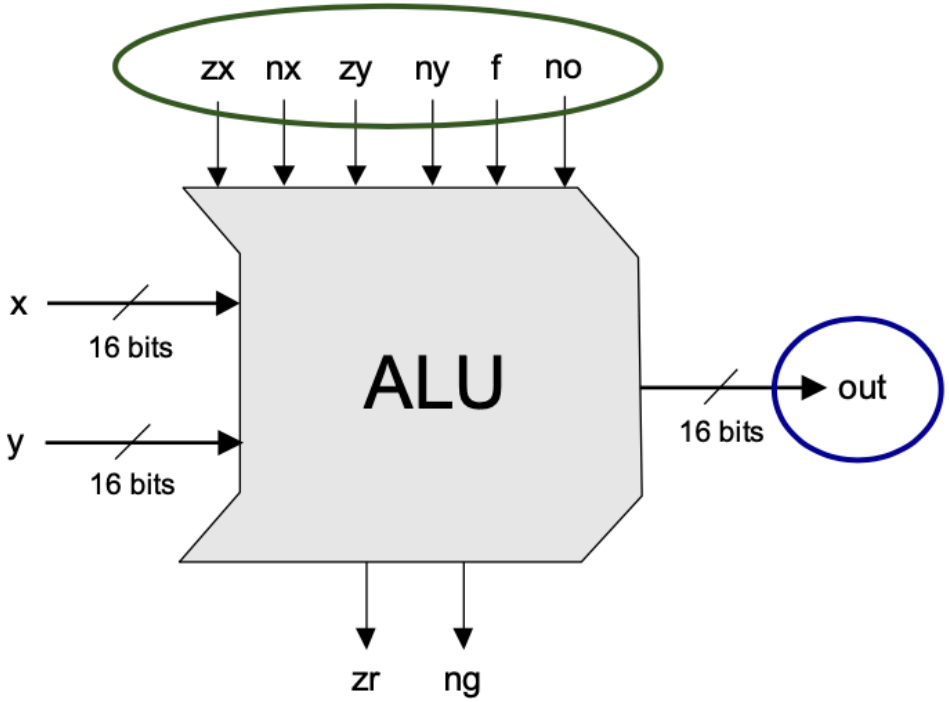
b12




```
And(a=instruction[15], b=instruction[12], out=cb4)
Mux16(a=oAReg, b=inM, sel=out=oMux2)
```

Control bit : cb5

- คำสั่ง ALU



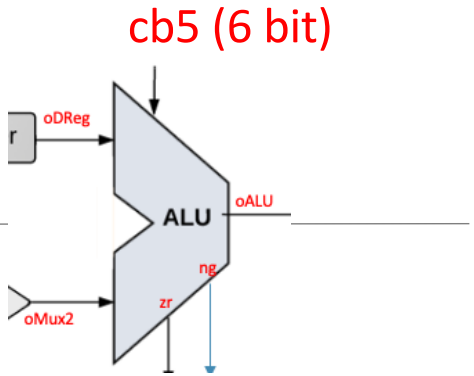
ALU instruction mapping

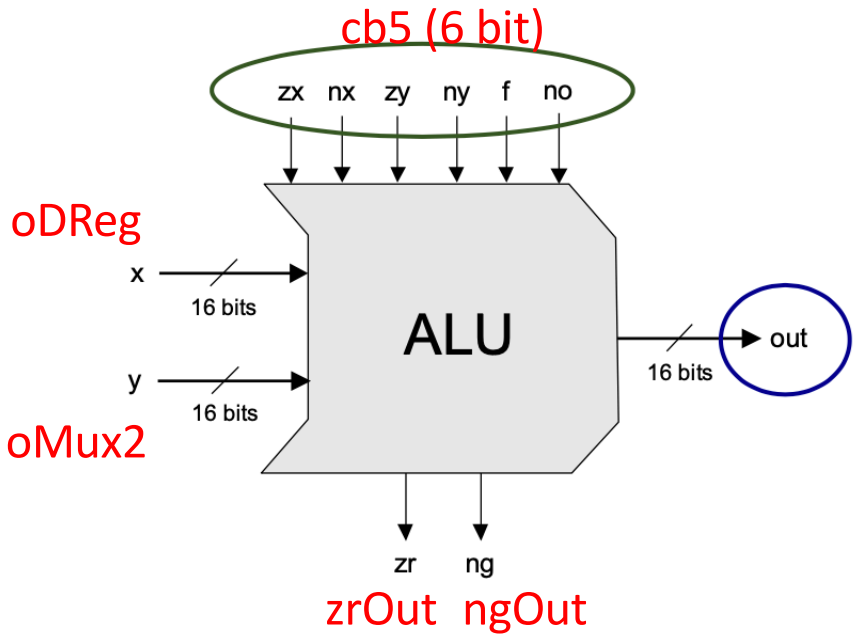
	comp		c1	c2	c3	c4	c5	c6
	0		1	0	1	0	1	0
	1		1	1	1	1	1	1
	-1		1	1	1	0	1	0
	D		0	0	1	1	0	0
	A	M	1	1	0	0	0	0
	!D		0	0	1	1	0	1
	!A	!M	1	1	0	0	0	1
x -	-D		0	0	1	1	1	1
	-A	-M	1	1	0	0	1	1
	D+1		0	1	1	1	1	1
y -	A+1	M+1	1	1	0	1	1	1
	D-1		0	0	1	1	1	0
	A-1	M-1	1	1	0	0	1	0
	D+A	D+M	0	0	0	0	1	0
	D-A	D-M	0	1	0	0	1	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						

zx	nx	zy	ny	f	no	out
1	0	1	0	1	0	0
1	1	1	1	1	1	1
1	1	1	0	1	0	-1
0	0	1	1	0	0	x
1	1	0	0	0	0	y
0	0	0	0	1	0	x+y
0	1	0	0	1	1	x-y
0	0	0	1	1	1	y-x
0	0	0	0	0	0	x&y
0	1	0	1	0	1	x|y

```
zx=instruction[11], //c1
nx=instruction[10], //c2
zy=instruction[9], //c3
ny=instruction[8], //c4
f=instruction[7], //c5
no=instruction[6], //c6
```

Control bit : cb5






```
x=oDReg, y=oMux2,
zx=instruction[11], //c1
nx=instruction[10], //c2
zy=instruction[9], //c3
ny=instruction[8], //c4
f=instruction[7], //c5
no=instruction[6], //c6
zr=zrOut, ng=ngOut,
out=oALU;
```

Control bit : cb6

- เกี่ยวข้องกับ j1 j2 j3 ใน C-instruction

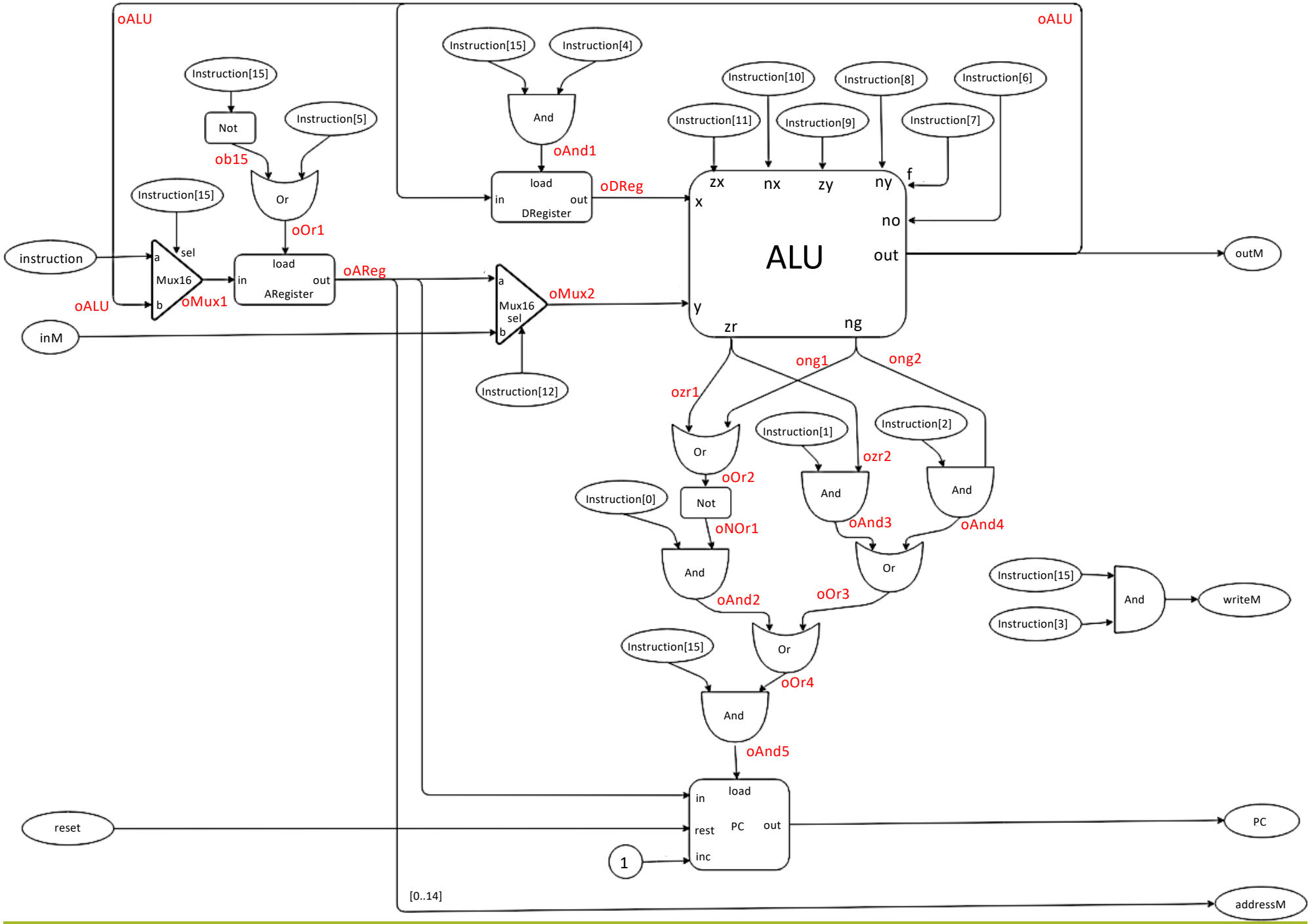
```
Not(in=ngOut, out=pos);                          // Reset: = 0
Not(in=zrOut, out=nzr);                          // 000 nojump: ++
And(a=instruction[15], b=instruction[0], out=jgt); // 111 goto: = A
And(a=pos, b=nzr, out=posnzr);                   // 010 100 etc conditional
And(a=jgt, b=posnzr, out=ld1);                   goto: = A || ++

And(a=instruction[15], b=instruction[1], out=jeq);
And(a=jeq, b=zrOut, out=ld2);

And(a=instruction[15], b=instruction[2], out=jlt);
And(a=jlt, b=ngOut, out=ld3);

Or(a=ld1, b=ld2, out=ldt);
Or(a=ld3, b=ldt, out=ld);

PC(in=oALU, load=ld, inc=true, reset=reset, out[0..14]=pc);
```



https://nihemak.hatenablog.com/entry/2019/04/28/150541

Coming up: W5.4 แฮกคอมพิวเตอร์