



## การจัดองค์การคอมพิวเตอร์

# Arithmetic Logic Unit

---

31110321 Computer Organization

สำหรับนักศึกษาชั้นปีที่ 3 สาขาวิชาวิศวกรรมคอมพิวเตอร์

ทรงฤทธิ์ กิติศรีวรพันธุ์

songrit@npu.ac.th

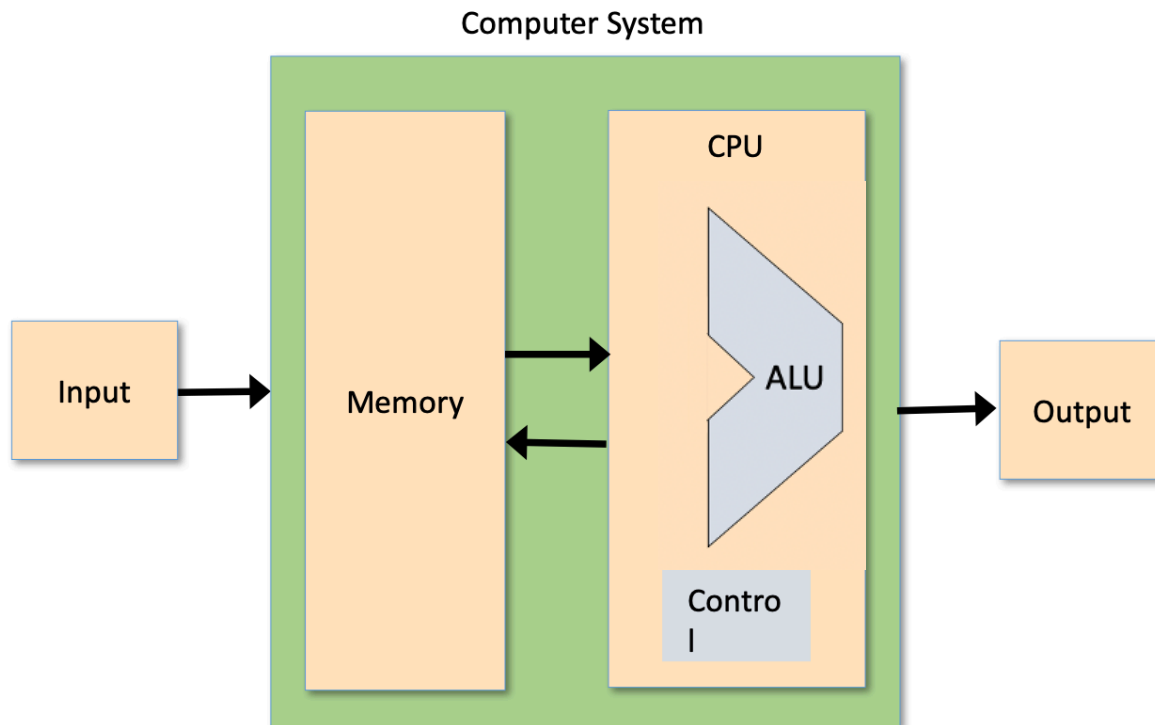
สาขาวิชาวิศวกรรมคอมพิวเตอร์  
มหาวิทยาลัยนครพนม

# Outline

---

- 2.1 เลขไบนารี
- 2.2 การบวกเลขไบนารี
- 2.3 ตัวเลขลบ ในระบบดิจิทัล
- **2.4 Arithmetic Logic Unit (ALU)**
- 2.5 โปรเจ็คสับดาห์ (พฤษภาคม)
- 2.6 ภาพรวม

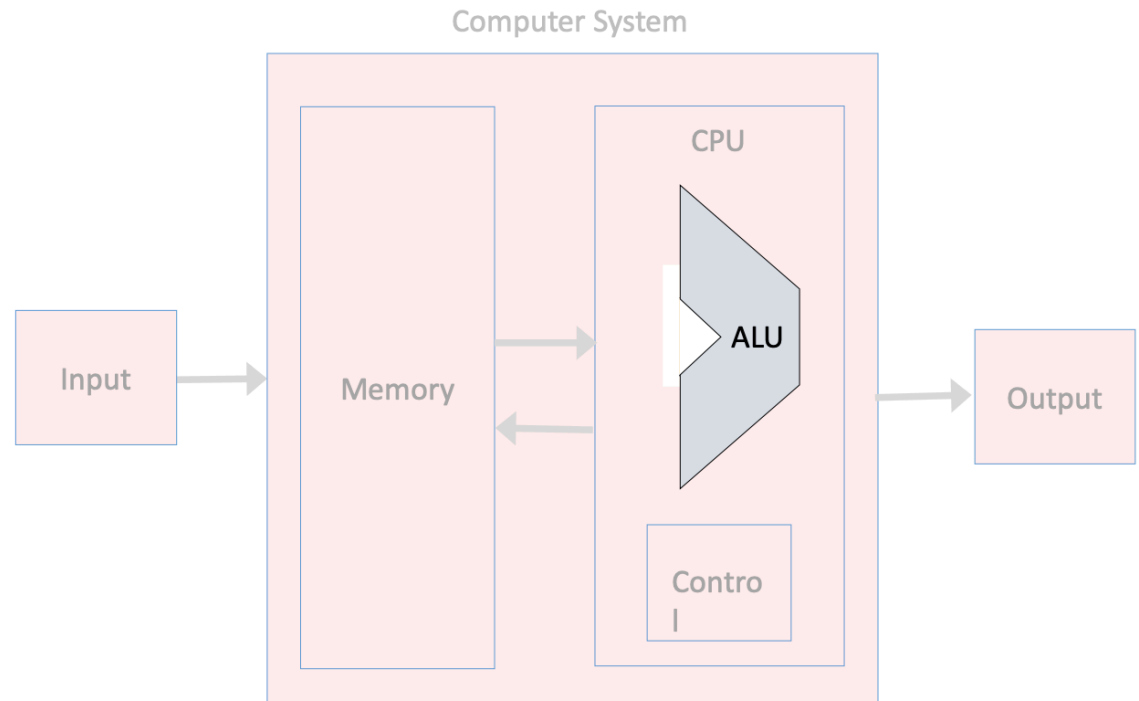
# Von Neumann Architecture



- 1945 John Von Neumann
- เสนอสถาปัตยกรรมคอมพิวเตอร์สำหรับ
- ใช้งานได้หลากหลาย

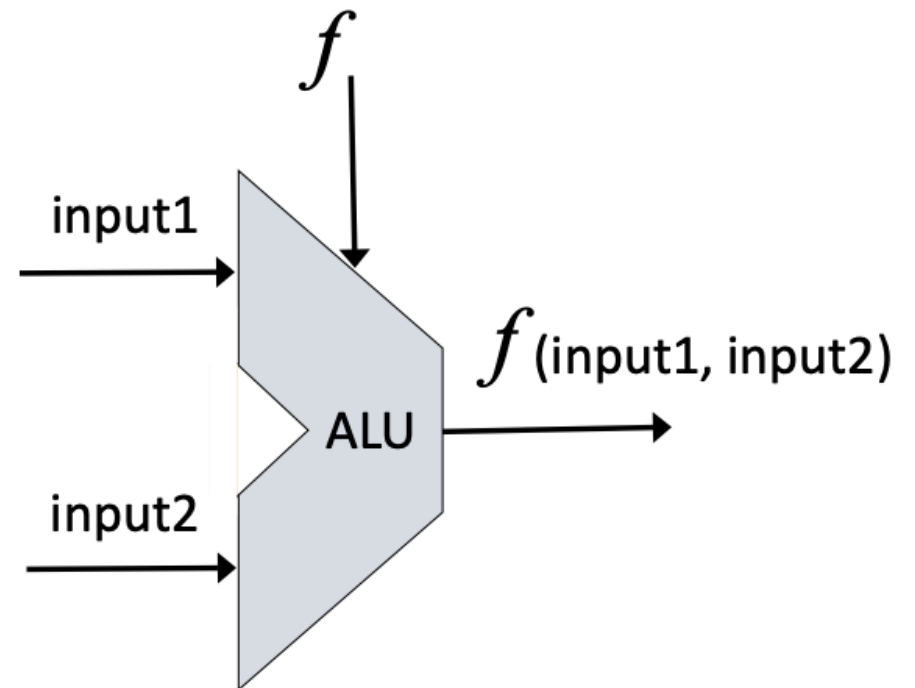
# ส่วนประมวลผลคณิตศาสตร์

- อุปกรณ์สำคัญ ของสถาปัตยกรรมฟอนนอยด์มัน คือ
- ส่วนประมวลผลคณิตศาสตร์
- หรือ เอแอลยู (ALU)



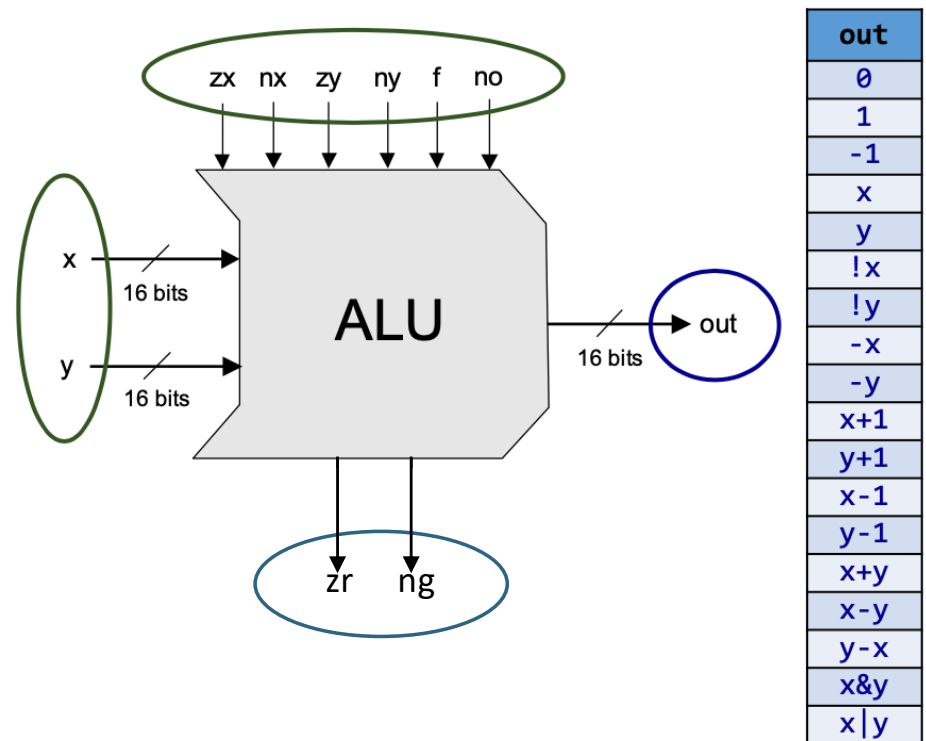
# Arithmetic Logical Unit

- ส่วนคำนวณเลขไบนารี มีการรับข้อมูล 2 ชุด เพื่อนำข้อมูลนั้นมาประมวลผลกัน
- ป้อนคำสั่งประมวลผลที่ตำแหน่ง  $f$ 
  - บวก
  - ลบ
  - เลื่อนบิต และ อื่น
- เมื่อคำนวณเสร็จส่งออก
  - $F(\text{input1}, \text{input2})$



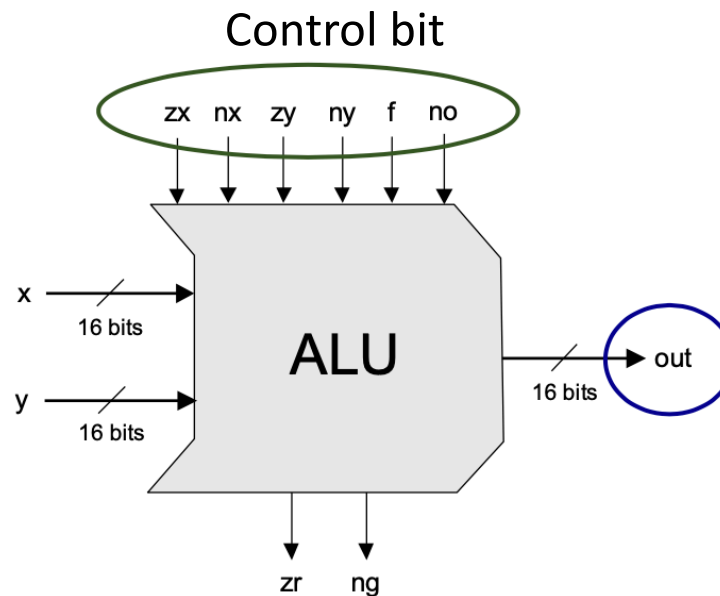
# Hack ALU

- คำนวณครั้งละ 16-bit
- คำนวณเลขลบ แบบ two's complement
- บัสอินพุต 16บิต 2 บัส
- มีคำสั่ง 6 บิต
- ส่งออกบัสเอาต์พุต 16บิต
- มี 2 บิตเอาต์พุต (zr, ng)



# Control Bit

- ALU ทำงานด้วยการ  
ส่งคำสั่งใน Control  
Bit
- เราได้ออกแบบคำสั่ง  
Control bit เป็น  
ตามตาราง



Control bit

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	1	1	0	1	!x
1	1	0	0	0	1	!y
0	0	1	1	1	1	-x
1	1	0	0	1	1	-y
0	1	1	1	1	1	x+1
1	1	0	1	1	1	y+1
0	0	1	1	1	0	x-1
1	1	0	0	1	0	y-1
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

# Hack ALU : คำนวณ $y-x$

The screenshot shows the Hack ALU simulator interface. The top toolbar contains icons for loading, running, and inspecting. The main window is divided into several sections: a top status bar, a left input/output table, a bottom HDL code editor, and a right output display area. Annotations in orange callouts provide instructions for using the simulator to calculate  $y-x$ .

**Annotations:**

- Load tools/builtInChips/ALU.hdl**: Points to the load icon in the toolbar.
- 2. Evaluate the chip logic**: Points to the run icon in the toolbar.
- 1. Set the ALU's inputs and control bits to some test values (000111 codes "output  $y-x$ ")**: Points to the input/output table where control bits are set to 1.
- 3. Inspect the ALU outputs**: Points to the output display area showing the result -10.
- Built-in ALU implementation**: Points to the HDL code editor showing the implementation of the ALU.
- The built-in ALU implementation has some GUI side-effects**: Points to the ALU block diagram in the bottom right corner.

**HDL Code (Bottom Left):**

```
// This file is part of the material for "The Elements of Computing Systems"
// by Noam Nisan and Shimon Rubinfeld. Copyright 2005 MIT Press. Book site: www.nand2tetris.org
// File name: tools/builtIn/ALU.hdl

/**
 * The ALU. Computes a pre-defined operation on two 16-bit integers x and y.
 * where x and y are two 16-bit integers.
 * by a set of 6 control bits described below.
 * The ALU operation can be described as follows:
 * * if zx=1 set x = 0
 * * if nx=1 set x = !x
 * * if zy=1 set y = 0
 * * if ny=1 set y = !y
 */
```

**Input/Output Table (Top Left):**

Name	Value
no	1

**Output Display (Top Right):**

Value
-10

**ALU Block Diagram (Bottom Right):**

ALU  
D Input : 30  
M/A Input : 20  
ALU output : -10



# Hack ALU : คำนวณ $x \& y$

The screenshot shows the Hack ALU simulator interface. The top menu bar includes File, View, Run, and Help. Below the menu is a toolbar with icons for running, stepping, and other functions. The main window is divided into several sections:

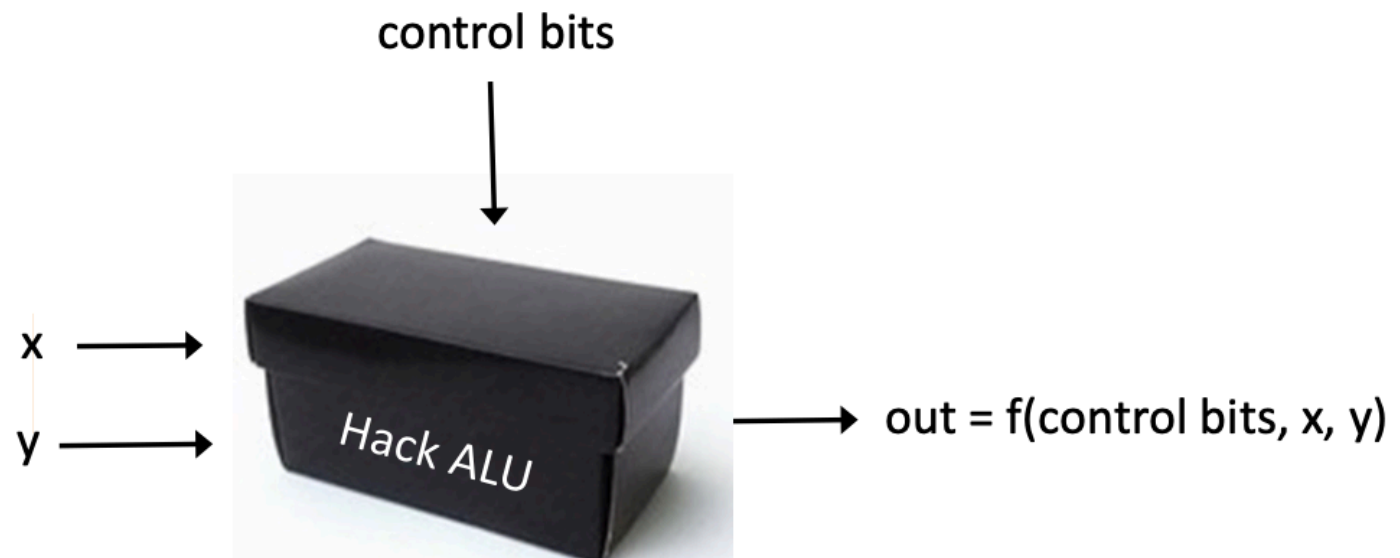
- Input pins:** A table showing the values of input pins. The values for `x[16]` and `y[16]` are circled in green. The values for `zx`, `nx`, `zy`, `ny`, `f`, and `no` are all 0.
- Output pins:** A table showing the values of output pins. The value for `out[16]` is circled in blue. The values for `zr` and `ng` are 0.
- HDL:** A text area showing the HDL code for the ALU. The code is commented out, showing the logic for computing `x & y`.
- ALU:** A diagram of the ALU component. It has two inputs: `D Input` (value -5242) and `M/A Input` (value 6253). The output is `ALU output` (value 2052).

Annotations with callouts provide additional information:

- Set to binary I/O format:** Points to the `Format` dropdown menu.
- Inspect the ALU outputs:** Points to the `out[16]` output pin value.
- Set the ALU's inputs and control bits to some test values (000000 codes "compute x&y"):** Points to the `zx`, `nx`, `zy`, `ny`, `f`, and `no` control bits.

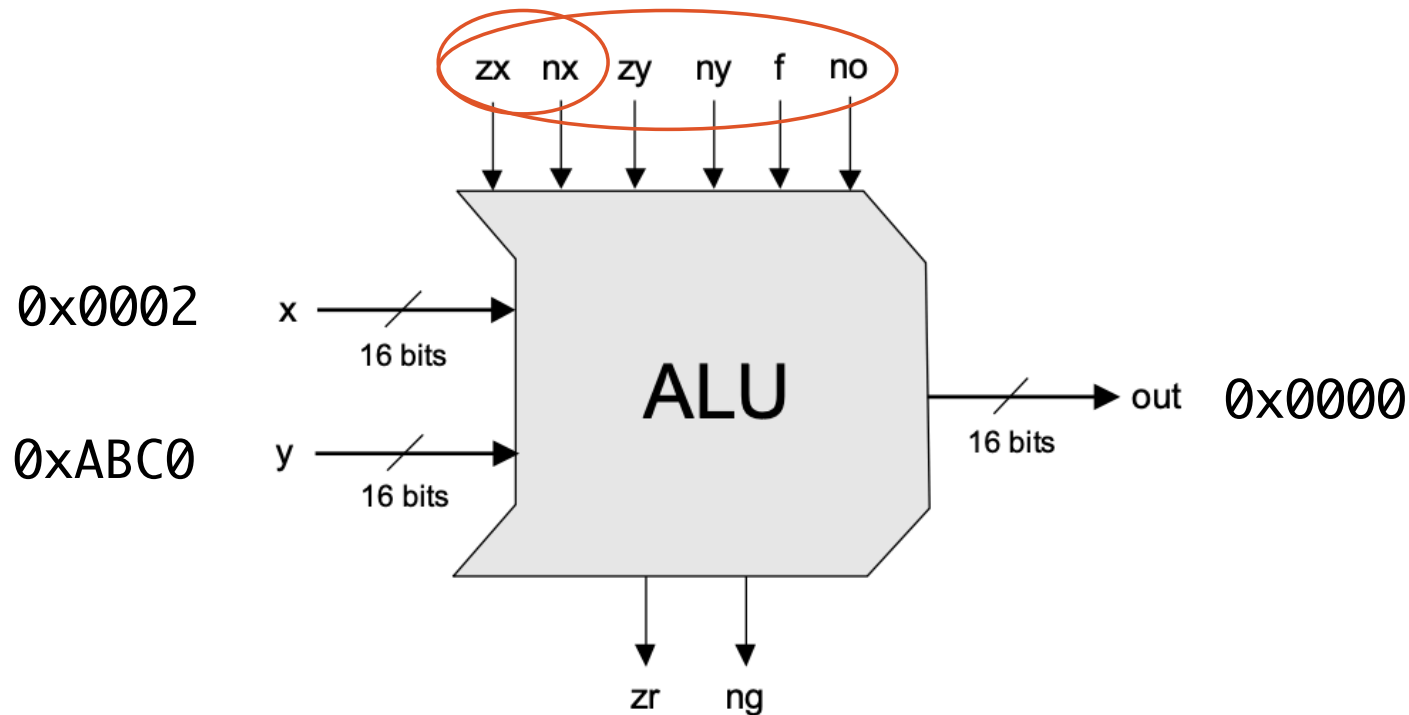
# Hack ALU black box

---



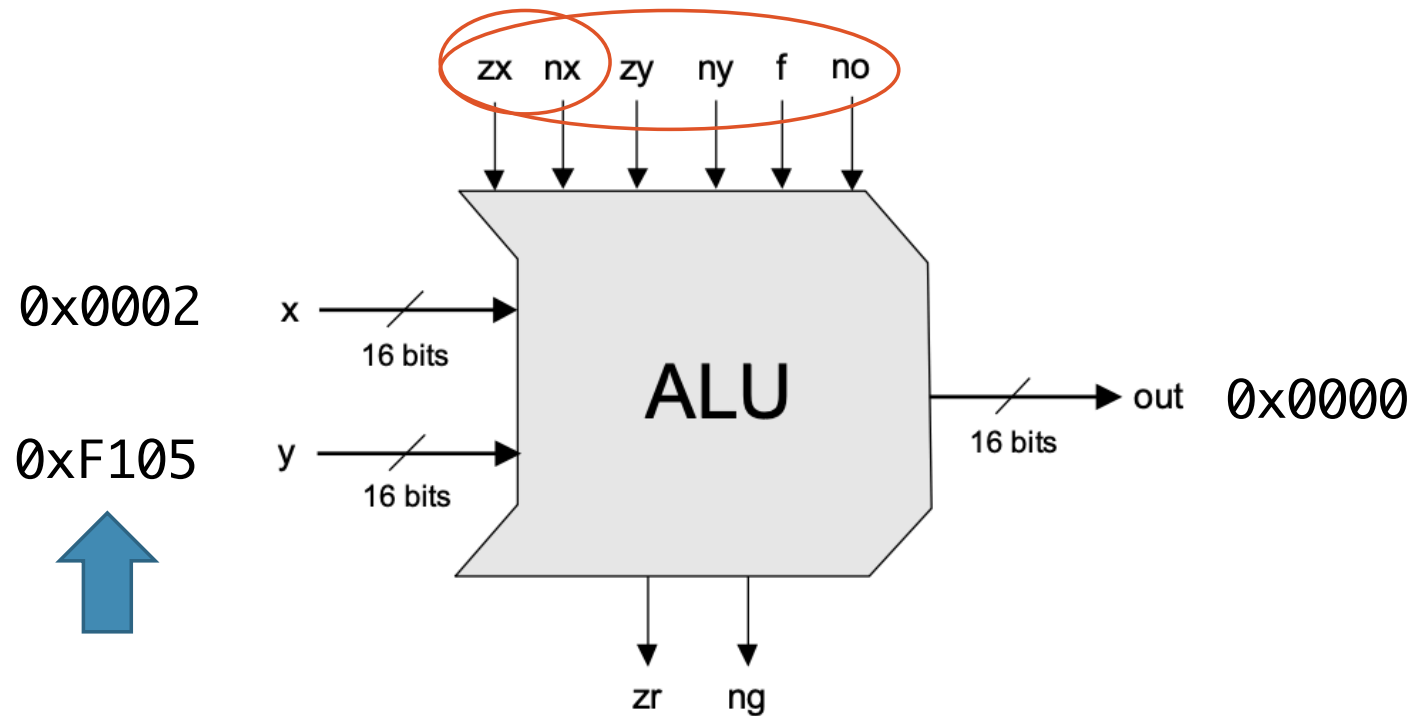
# The Hack ALU operation

zx	nx	zy	ny	f	no	out
if zx then x=0	if nx then x=!x					



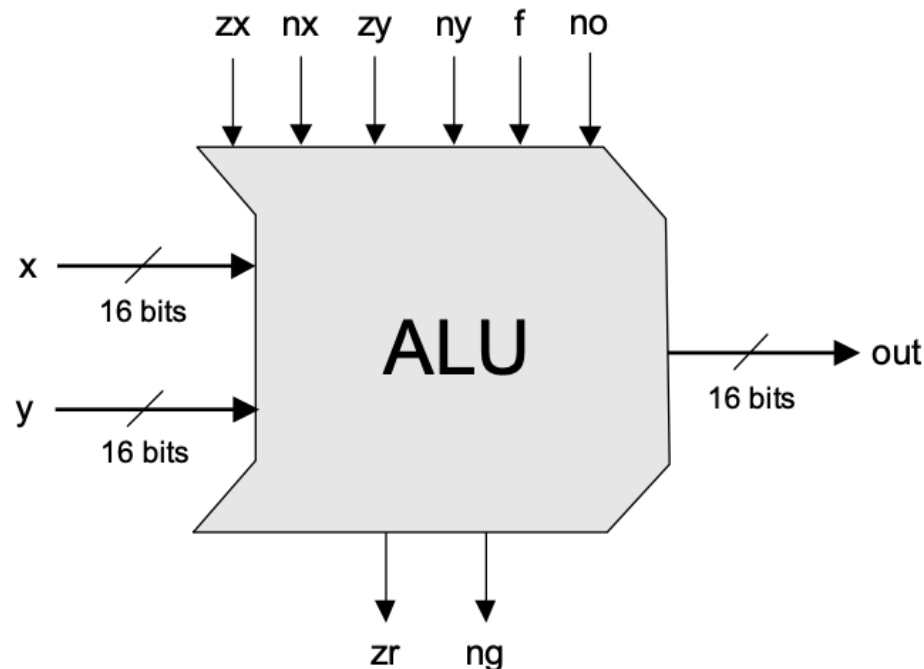
# The Hack ALU operation

zx	nx	zy	ny	f	no	out
if zx then x=0	if nx then x=!x					



# Hack ALU operation

pre-setting the x input		pre-setting the y input		selecting between computing + or &	post-setting the output	Resulting ALU output
zx	nx	zy	ny	f	no	out
if zx then x=0	if nx then x=!x	if zy then y=0	if ny then y=!y	if f then out=x+y else out=x&y	if no then out=!out	out(x,y)=



# Hack ALU operation

- !x
- inverse x

pre-setting the x input		pre-setting the y input		selecting between computing + or &	post-setting the output	Resulting ALU output
zx	nx	zy	ny	f	no	out
if zx then x=0	if nx then x=!x	if zy then y=0	if ny then y=!y	if f then out=x+y else out=x&y	if no then out=!out	out(x,y)=
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	1	1	0	1	!x
1	1	0	0	0	1	!y
0	0	1	1	1	1	-x
1	1	0	0	1	1	-y
0	1	1	1	1	1	x+1
1	1	0	1	1	1	y+1
0	0	1	1	1	0	x-1
1	1	0	0	1	0	y-1
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

Example: compute !x

x:

y:

Following pre-setting:

x:

y:

Computation and post-setting:

# Hack ALU

Example: compute y-x

x:

y:

Following pre-setting:

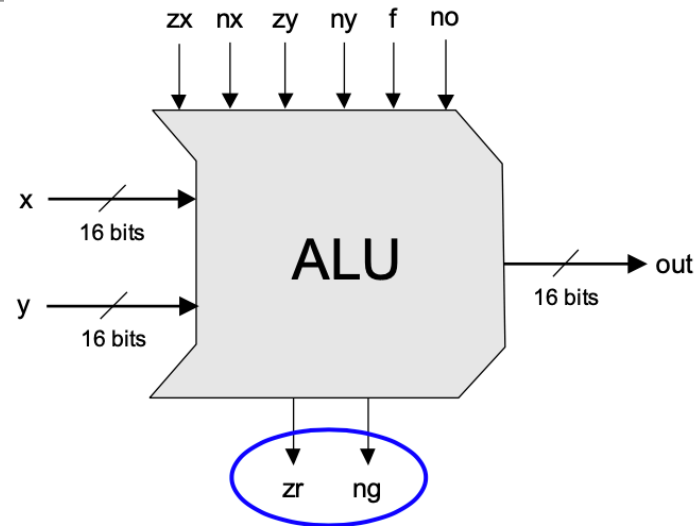
x:

y:

Computation and post-setting:

pre-setting the x input		pre-setting the y input		selecting between computing + or &	post-setting the output	Resulting ALU output
zx	nx	zy	ny	f	no	out
if zx then x=0	if nx then x=!x	if zy then y=0	if ny then y=!y	if f then out=x+y else out=x&y	if no then out=!out	out(x,y)=
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	1	1	0	1	!x
1	1	0	0	0	1	!y
0	0	1	1	1	1	-x
1	1	0	0	1	1	-y
0	1	1	1	1	1	x+1
1	1	0	1	1	1	y+1
0	0	1	1	1	0	x-1
1	1	0	0	1	0	y-1
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

# Hack ALU output control bits



- if (out == 0) then zr = 1, else zr = 0
- if (out < 0) then ng = 1, else ng = 0
- ได้ใช้ zr และ ng เมื่อขั้นตอนการสร้าง
  - แอ็กคัมพิวเตอรื ในบทที่ 4



# Coming up: W2.5 Project

---

ภาพรวม โปรเจ็คประจำสัปดาห์