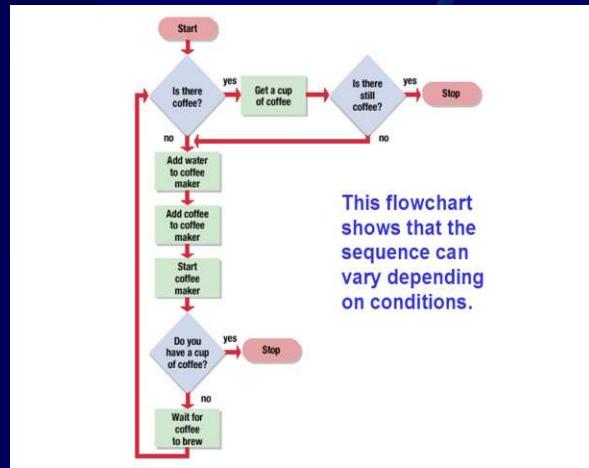


# Hardware, Software, Flowchart



This flowchart  
shows that the  
sequence can  
vary depending  
on conditions.

Good students never miss a class !!!



# เนื้อหา

- อุปกรณ์ Hardware
- ประเภทของ Software
- ขั้นตอนการพัฒนาโปรแกรม
- การวิเคราะห์โจทย์ปัญหา
- การเขียนผังงาน
- การ Compile โปรแกรมภาษา C

# 1.1 อุปกรณ์ Hardware

คอมพิวเตอร์ หมายถึง เครื่องมือทาง  
อิเล็กทรอนิกส์ ที่ใช้ในการประมวลผลข้อมูล

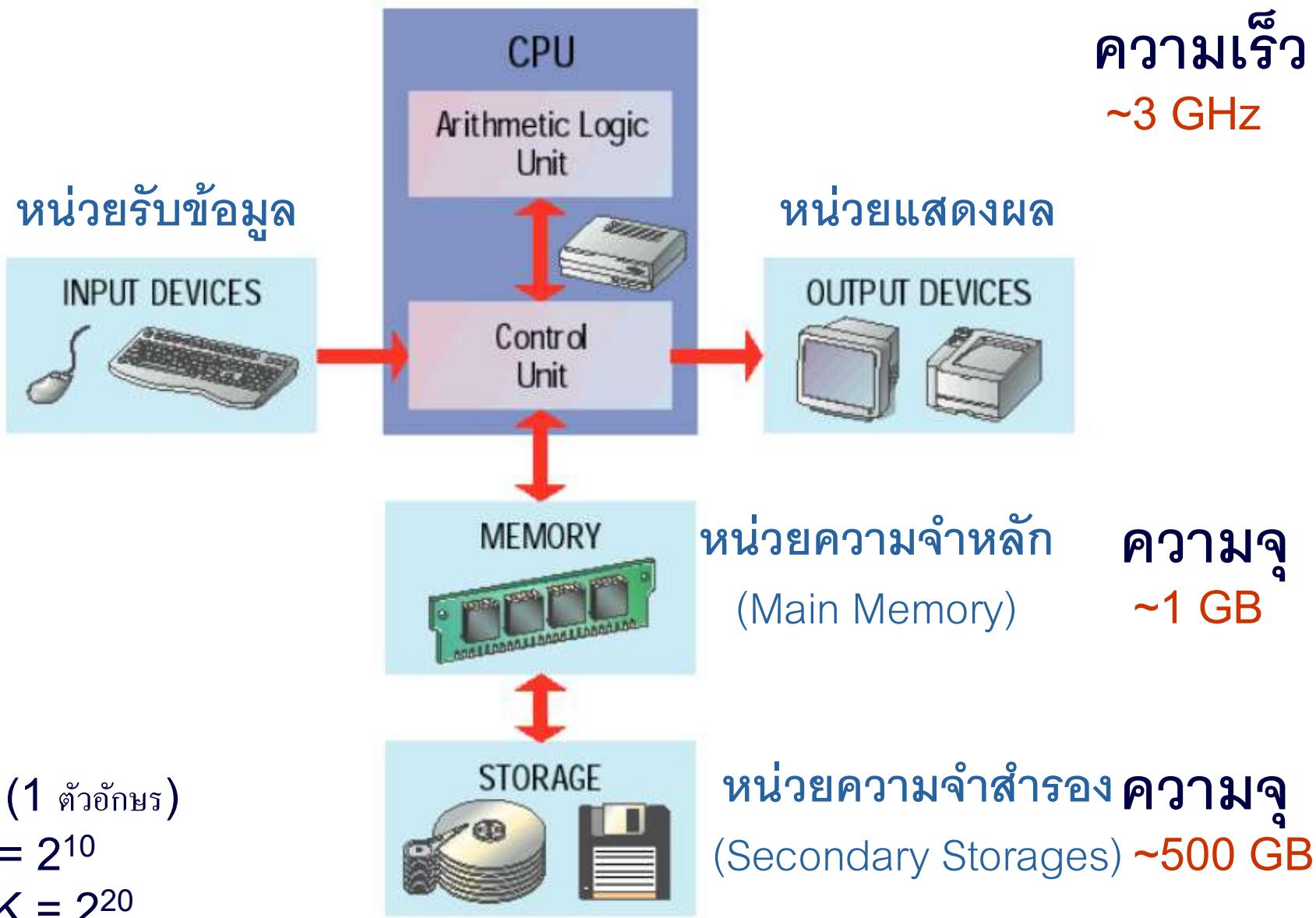
ส่วนประกอบหลักของคอมพิวเตอร์

- หน่วยประมวลผลกลาง (CPU)
- หน่วยความจำหลัก (Main Memory)
- หน่วยความจำสำรอง (Secondary Storages)
- หน่วยรับและแสดงผล (Input/Output)



# หน่วยประมวลผลกลาง

(Central Processing Unit)



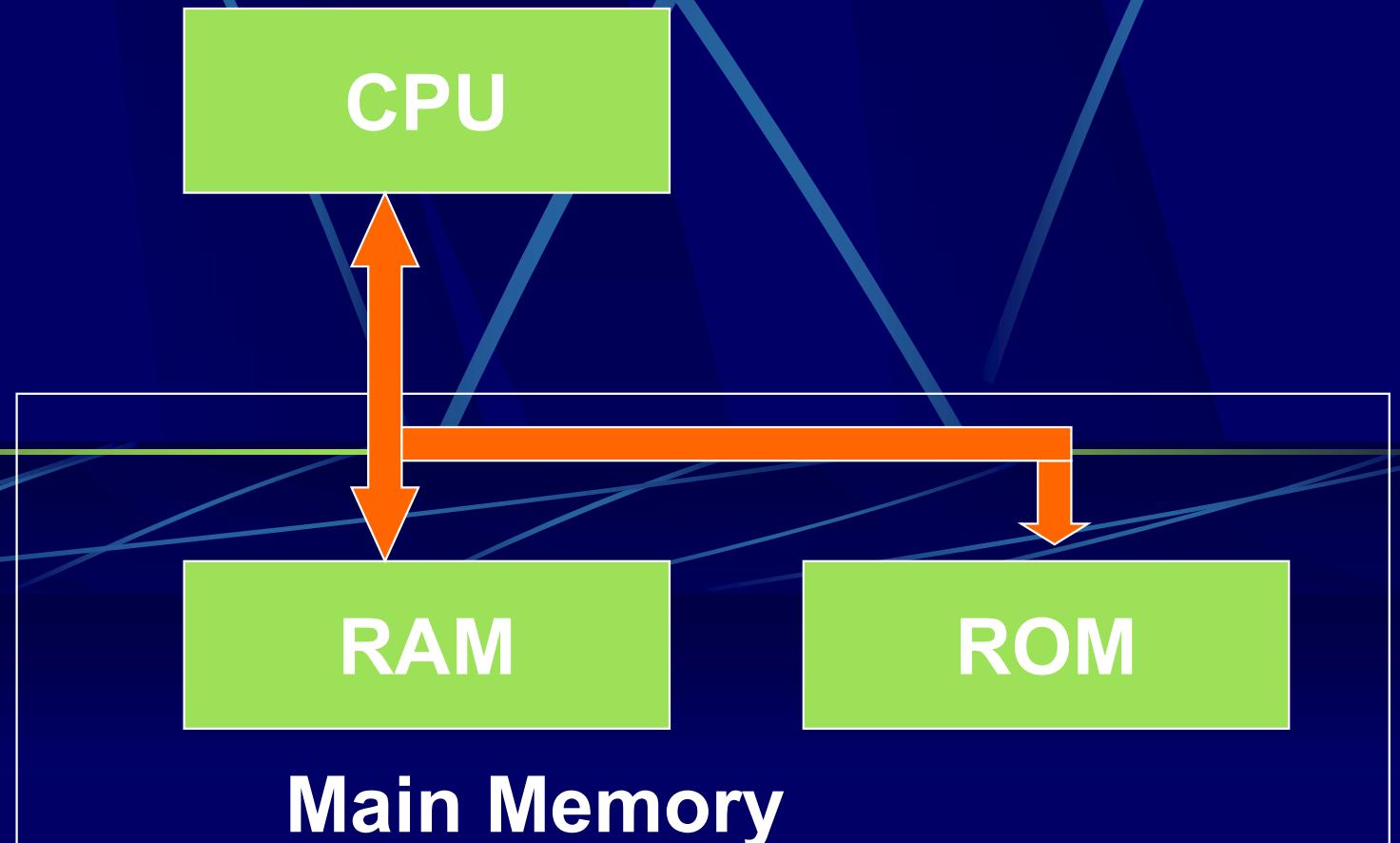
1B = 8 bits (1 ตัวอักษร)

1K = 1024 =  $2^{10}$

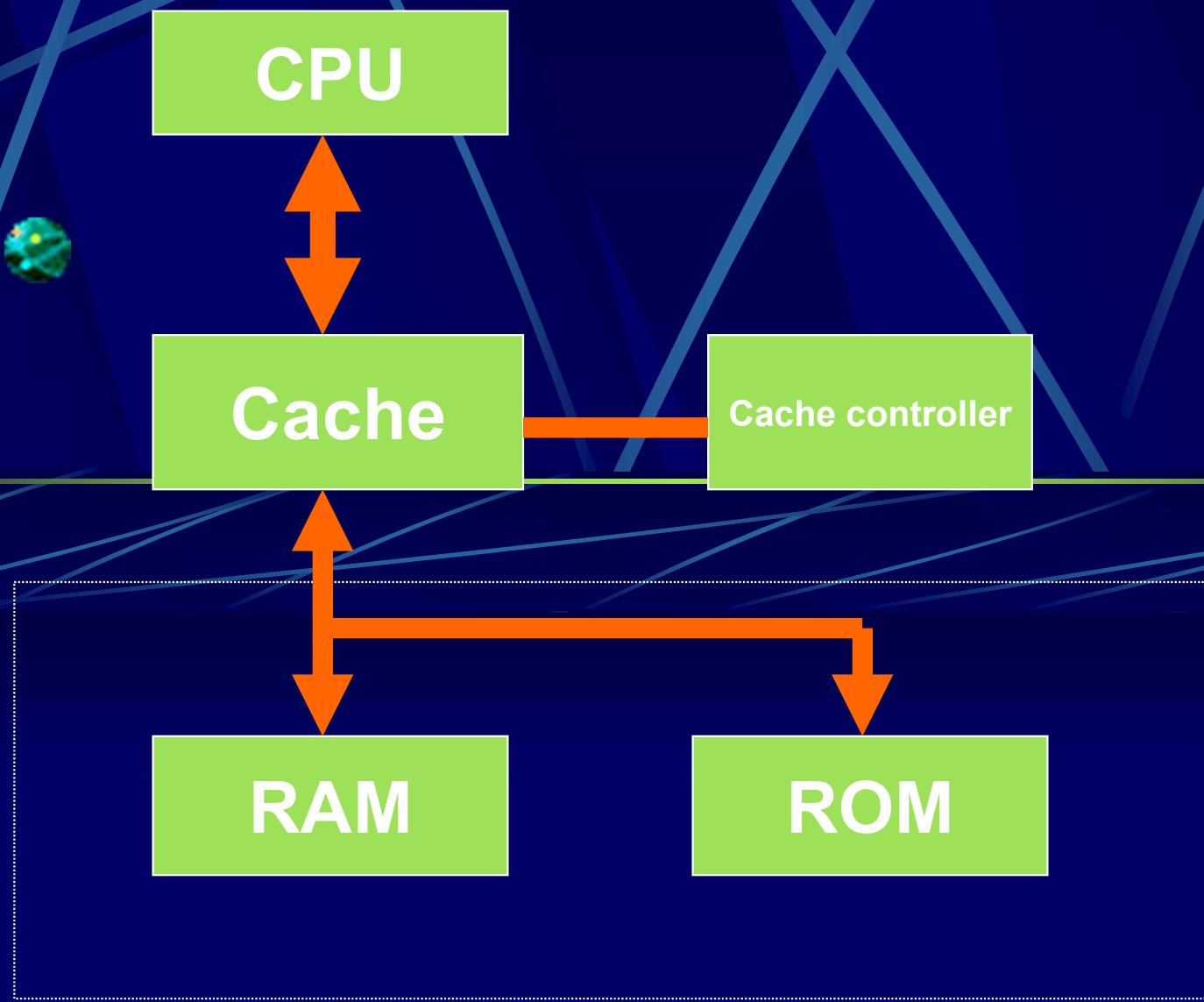
1M = 1024K =  $2^{20}$

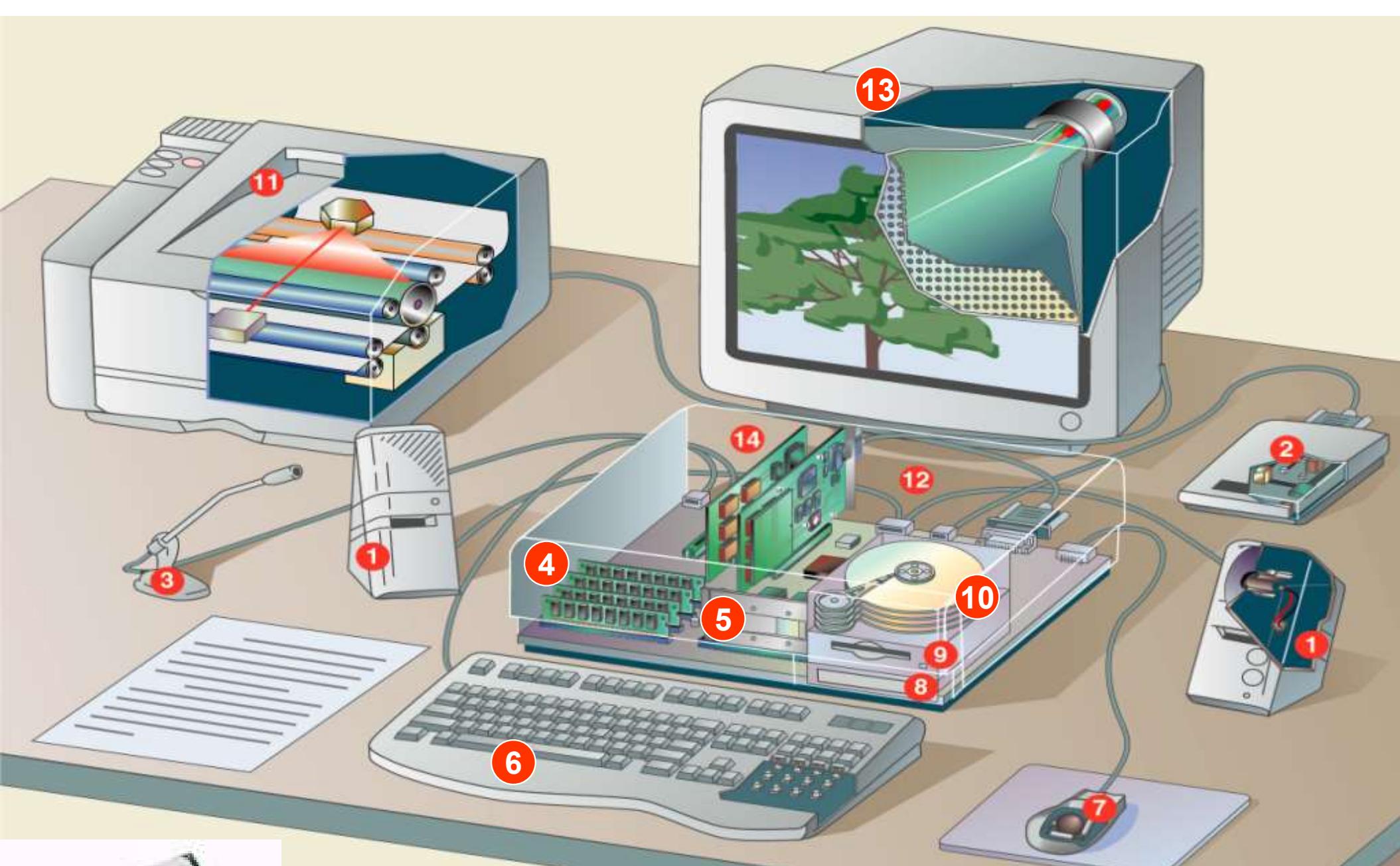
1G = 1024M =  $2^{30}$

# ໂຄຣງສ້າງບັນເມນບອໍດ



# โครงสร้างบันเม้นบอร์ด





intel®  
pentium® 4

- 1 Speakers
- 2 Modem
- 3 Microphone
- 4 RAM (Main memory)
- 5 CPU
- 6 Keyboard

- 7 Mouse
- 8 CD-ROM drive
- 9 Diskette drive
- 10 Hard drive
- 11 Printer
- 12 Ports
- 13 Monitor
- 14 Expansion board

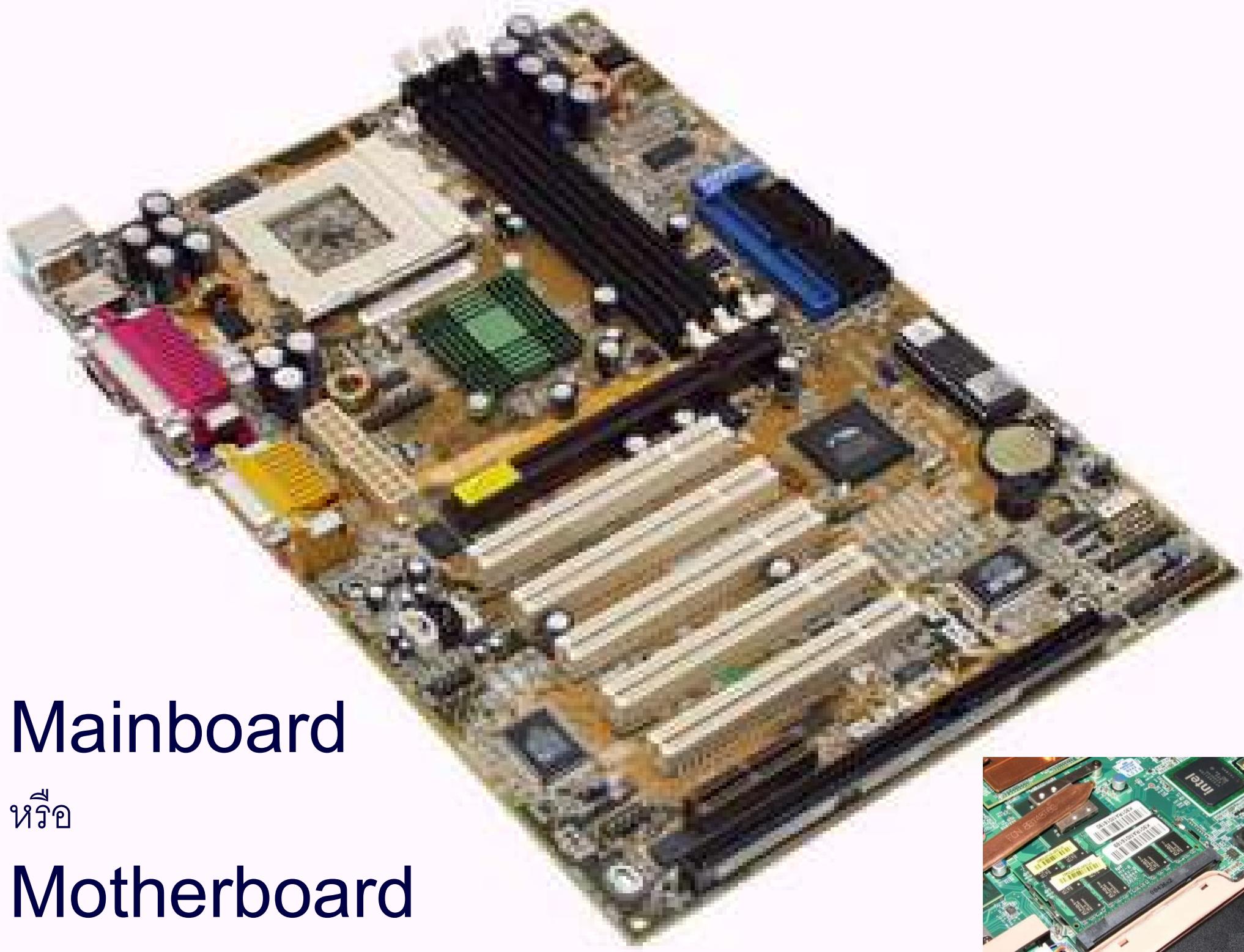
# 1.1.1 CPU

## ● หน่วยประมวลผลกลาง (Central Processing Unit)

- CPU ทำหน้าที่หลักในการประมวลผลข้อมูล และควบคุมการทำงานทั้งหมดของระบบ
- เป็นเสมือนสมองของคอมพิวเตอร์ในส่วนคำนวณ

## ● หน่วยประมวลผล (CPU) ใน PC

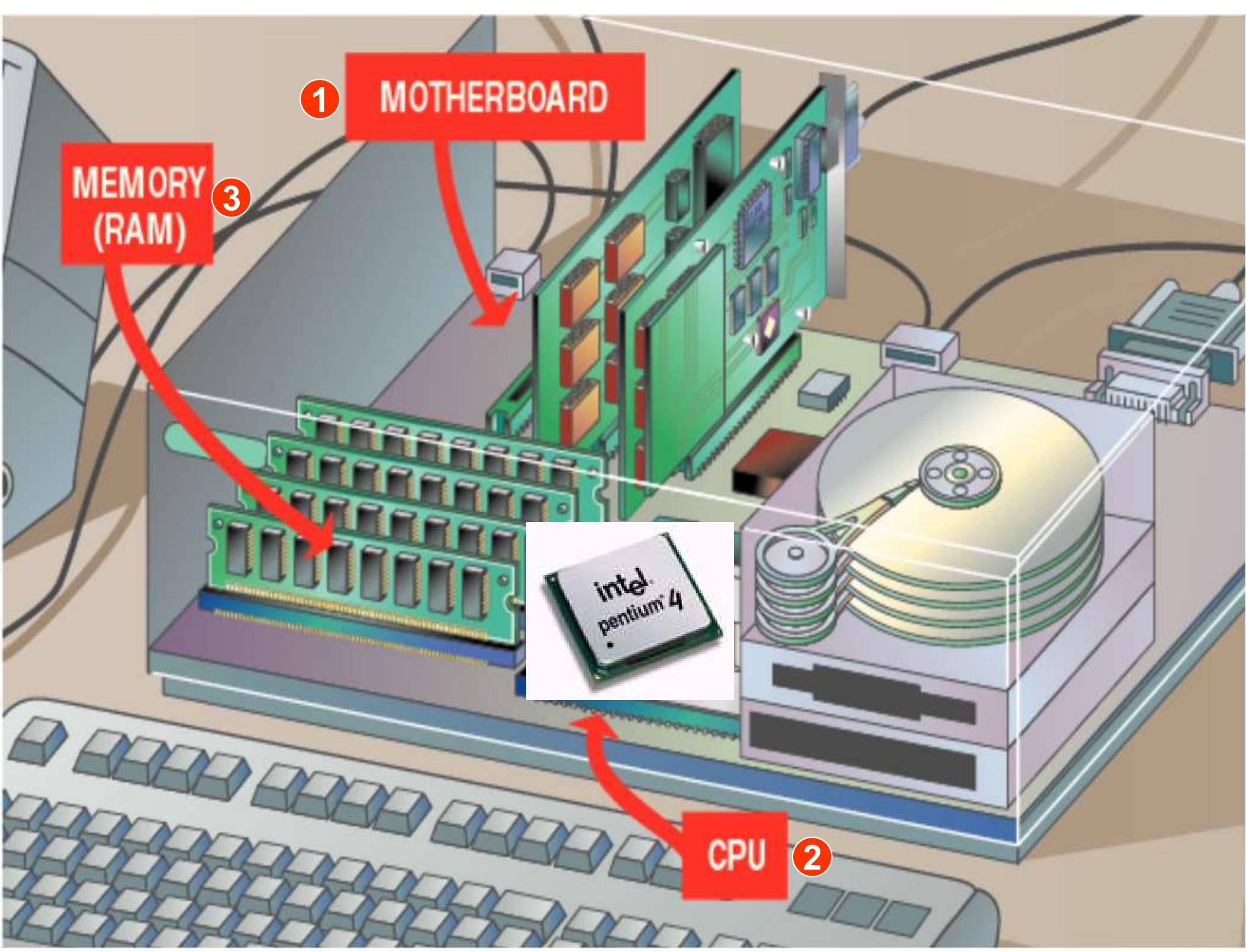
- เป็นหน่วยประมวลผลขนาดเล็กเรียกว่า Microprocessor วางอยู่บน แผงวงจรหลัก (Main Board)
- ซึ่งประกอบด้วยวงจร ที่เชื่อมต่อระหว่างหน่วยประมวลผล (CPU) กับหน่วยอื่นๆ (Memory, Disk, I/O)



# Mainboard

hw

# Motherboard



## 1.1.2 หน่วยความจำหลัก

### หน่วยความจำหลัก (Main Memory)

- เป็นที่เก็บโปรแกรมและข้อมูลที่อยู่ในระหว่างการประมวลผล
- โดยเก็บใน RAM (Random Access Memory) แบบชั่วคราว
- ขนาดของ RAM (2-16 GB) แสดงถึงประสิทธิภาพของคอมพิวเตอร์

### ความจุของหน่วยความจำ มีหน่วยวัด ดังนี้

- 1 kB (Kilobytes) =  $2^{10}$  = 1024 Bytes
- 1 MB (Megabytes) =  $2^{20}$  = 1024 kB (= 1,048,576 Bytes)
- 1 GB (Gigabytes) =  $2^{30}$  = 1024 MBs
- 1 TB (Terabytes) =  $2^{40}$  = 1024 GBs

# 1.1.3 หน่วยความจำสำรอง

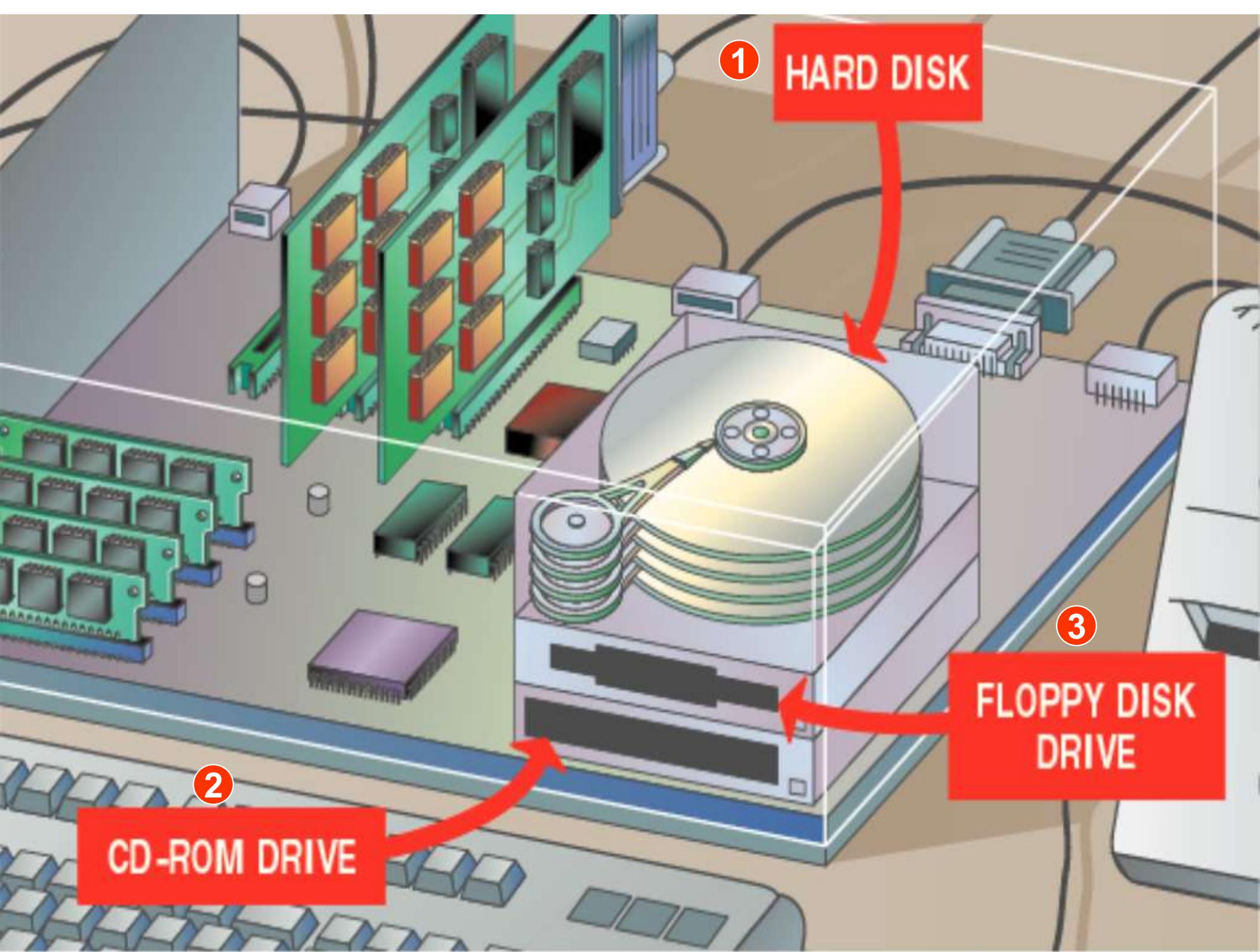
## หน่วยความจำสำรอง (Secondary Storages)

- เก็บโปรแกรมและข้อมูลแบบถาวร
- ทั้งที่กำลังประมวลผลและยังไม่ถูกประมวลผลในขณะนั้น

## ชนิดของอุปกรณ์ ที่ใช้เป็นหน่วยความจำสำรอง

- Hard drive
- Floppy disk drive
- CD/DVD-ROM drive, ...





## 1.1.4 หน่วยรับ/แสดงผล

● หน่วยรับ/แสดงผล (Input/Output)

● อุปกรณ์รับข้อมูล (Input Devices)

- เป็นเครื่องมือในการรับข้อมูล และคำสั่งจากผู้ใช้
- เช่น Keyboard, Mouse, Microphone

● อุปกรณ์แสดงผลข้อมูล (Output Devices)

- เป็นเครื่องมือในการส่งผลการทำงานกลับมายังผู้ใช้
- เช่น จอภาพ เครื่องพิมพ์ ลำโพง



# 1.2 ประเภทของ Software

● **Software** คือโปรแกรมหรือชุดคำสั่ง ที่ผู้เขียนสร้างขึ้น เพื่อให้คอมพิวเตอร์ทำงานอย่างเป็นขั้นตอน และได้ผลลัพธ์ตามที่ต้องการ

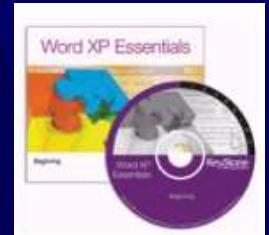
● **Software** แบ่งได้ 2 ประเภท

1. ซอฟต์แวร์ระบบ (Operating System: OS)

- เช่น DOS, Windows, UNIX, ...

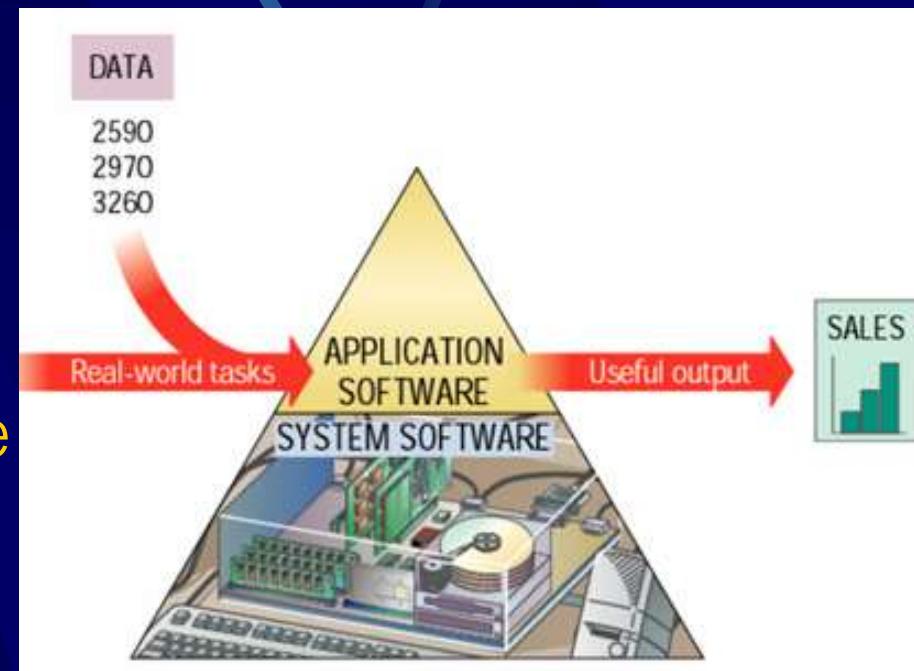
2. ซอฟต์แวร์ประยุกต์ (Application Program)

- **Software package** เช่น Word, Spreadsheet, ...
- **Developed program** เช่น โปรแกรมใช้งานเฉพาะ ถูกพัฒนาโดยใช้ โปรแกรมภาษา (Programming language เช่น Pascal, C, ...) 15



## 1.2.1 OS

- โปรแกรม OS ทำหน้าที่ในการควบคุมการทำงานของระบบคอมพิวเตอร์ เพื่อให้อุปกรณ์ต่างๆทำงานร่วมกันอย่างมีประสิทธิภาพ ตั้งแต่เริ่มเปิดเครื่อง
- เป็นโปรแกรมสื่อกลางระหว่างอุปกรณ์ Hardware และโปรแกรมประยุกต์ Software
- OS ที่นิยมใช้ เช่น



- Windows (สำหรับ PCs), UNIX (สำหรับ Minicomputers)

## 1.2.2 โปรแกรมภาษา

● **โปรแกรมภาษา** ใช้ในการพัฒนาโปรแกรม  
ประยุกต์สำหรับงานเฉพาะตามที่ผู้ใช้ต้องการ

● **ประเภทของโปรแกรมภาษา**

1. **ภาษาระดับต่ำ** (Low-Level Language)

เช่น ภาษาเครื่อง (Machine language)

2. **ภาษาระดับกลาง** (Middle-Level Language)

เช่น ภาษาแอสเซมบลี (Assembly Language)

3. **ภาษาระดับสูง** (High-Level Language)

เช่น Pascal, Fortran, C, JAVA, ...

## 1.2.2.1 ภาษาเครื่อง

● **ภาษาเครื่อง (Machine Language)**

เป็นภาษาที่คอมพิวเตอร์เข้าใจ ซึ่งเขียนเป็นรหัสเลขฐาน 2 (0/1) และคำสั่งมีความเกี่ยวข้องกับอุปกรณ์ของคอมพิวเตอร์โดยตรง

● **แต่มนุษย์เข้าใจภาษาเครื่องได้ยาก ดังนั้นการเขียนโปรแกรมด้วยภาษาเครื่องจึงยากมาก**

```
00010100101101010101010101010101010100010  
11101101010101010101010110010100010110  
00101001010100101111010111010111010101010  
10010100101101010101010101010101010110110  
011010010011001011110101110101010100010  
0001000101011101010100010101010111010  
10101001010100101011010111010111010111011  
000101001011010101010101010101010100010
```

## Object code

# หุ้นยนต์ของฉัน

00

เดินหน้า

01

ยกมือขึ้น

10

ส่งเสียง

11

ถอยหลัง

นำมาทำเป็นโปรแกรม



00

00

01

10

## 1.2.2.2 ภาษาแอสเซมบลี

### ภาษาแอสเซมบลี (Assembly Language)

เป็นภาษาที่เขียนโดยใช้คำสั่งที่มีนุ้ยญ์เข้าใจ (English-like statements) และการใช้รหัสเลขฐาน 2

- แต่ออกแบบมาเฉพาะสำหรับคอมพิวเตอร์แต่ละแบบ
- และผู้เขียนโปรแกรมยังต้องทราบข้อมูลที่เกี่ยวข้องกับอุปกรณ์ของคอมพิวเตอร์

### ใช้ แอสเซมเบอร์ (Assembler) ในการ

แปลภาษาแอสเซมบลี ให้เป็นภาษาเครื่อง

```
;CLEAR SCREEN USING BIOS
CLR: MOV AX,0600H      ;SCROLL SCREEN
      MOV BH,30        ;COLOUR
      MOV CX,0000        ;FROM
      MOV DX,184FH       ;TO 24,79
      INT 10H          ;CALL BIOS;

;INPUTTING OF A STRING
KEY: MOV AH,0AH        ;INPUT REQUEST
      LEA DX,BUFFER     ;POINT TO BUFFER WHERE STRING STORED
      INT 21H          ;CALL DOS
      RET              ;RETURN FROM SUBROUTINE TO MAIN PROGRAM;

; DISPLAY STRING TO SCREEN
SCR: MOV AH,09H        ;DISPLAY REQUEST
      LEA DX,STRING     ;POINT TO STRING
      INT 21H          ;CALL DOS
      RET              ;RETURN FROM THIS SUBROUTINE;
```

## Assembly code

Assembler

000101001011010101010101010101010100010  
1110110101010101010101110010100010110  
0010100101010010111101011101011101010  
1001010010110101010101010101010110110110  
0110100100110010111101011101010100010  
0001000101011101010100010101010111010  
1010100101010010101101011101011101011101011  
0001010010110101010101010101010100010

Object code

## 1.2.2.3 ภาษาระดับสูง

● **ภาษาระดับสูง (High-Level Languages)**

ใช้ภาษาที่มนุษย์เข้าใจ (English-like language)

เช่น Basic, Pascal, C, JAVA, ...

● **คอมไพล์เตอร์ (Compiler) หรือ Interpreter**

ในการแปลภาษาระดับสูงให้เป็นภาษาเครื่อง

● Compiler แปลทั้งโปรแกรม (เช่น Pascal, C, ...)

● Interpreter แปลทีละบรรทัด (เช่น Basic, ...) 23

```
#include <stdio.h>
#include <math.h>
void main()
{ const float pi = 3.1415926;
  float radius = 2.0, area, radius_check;
  printf("Input: Enter radius > ");
  scanf("%f", &area);
  area = pi * radius * radius;
  printf("Output: radius = %f\n", radius);
  printf("           area    = %f\n", area);
}
```

## C code

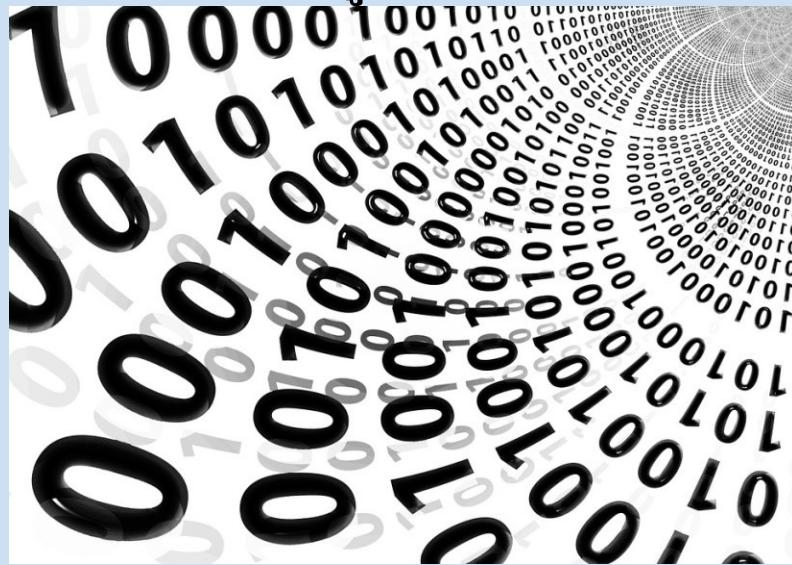
Compiler

```
000101001011010101010101010101010100010
1110110101010101010101110010100010110
0010100101010010111101011101011101010
1001010010110101010101010101010110110110
0110100100110010111101011101010100010
0001000101011101010101000101010111010
1010100101010010101101011101011101011101011
000101001011010101010101010101010100010
```

Object code

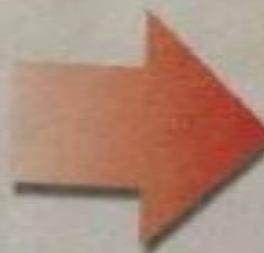
# รหัสแทนข้อมูล

- เนื่องจากคอมพิวเตอร์ไม่สามารถเข้าใจภาษามนุษย์ได้ จึงต้องมีรหัสที่ใช้แทนข้อมูล เพื่อสั่งให้คอมพิวเตอร์ทำงาน
- รหัสแทนข้อมูลที่นิยมใช้ได้แก่ ASCII หรือ รหัสแอลกิ



### ขั้นตอนที่ 1

กด **Shift** + **D** เพื่อป้อนตัวอักษร “D”

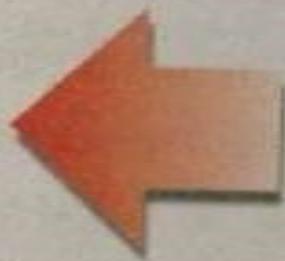


### ขั้นตอนที่ 2

สัญญาณอักษร “D”

ส่งต่อไปยังระบบการทำงาน  
ของคอมพิวเตอร์

D



### ขั้นตอนที่ 3

แปลงตัวอักษร “D” ให้อยู่ใน  
รูปแบบมาตรฐานของรหัส ASCII

ตัวอักษร ASCII	Binary	Decimal	Octal	Hex
A	01000001	65	101	41
B	01000010	66	102	42
C	01000011	67	103	43
D	01000100	68	104	44
E	01000101	69	105	45
F	01000110	70	106	46
G	01000111	71	107	47
H	01001000	72	110	48
I	01001001	73	111	49
J	01001010	74	112	4A
K	01001011	75	113	4B
L	01001100	76	114	4C
M	01001101	77	115	4D
N	01001110	78	116	4E
O	01001111	79	117	4F



### ขั้นตอนที่ 4

แสดงผล โดยแปลงกลับ  
ให้เป็นภาษาอักษร “D”

บนอุปกรณ์แสดงผล

01000100

## ตัวอย่าง

ข้อความว่า BANGKOK

010000100100000101001  
110010001110100101101  
00111101001011

	หน่วยความจำ
B	01000010
A	01000001
N	01001110
G	01000111
K	01001011
O	01001111
K	01001011

# ตัวอย่าง

ข้อความว่า นนทบุรี

	b7	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1
	b6	0	0	0	0	1	1	1	1	0	0	0	0	1	1	1
	b5	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1
	b4	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0
b3	b2	b1	b0													
0	0	0	0	0	@	P	'	p	ສ	ກ	ຂ	ີ	ອ			
0	0	0	1	!	1	A	Q	a	ກ	ມ	ດ	ະ	ເ			
0	0	1	0	*	2	B	R	b	r	ຂ	ພ	ກ	ີ	ເ		
0	0	1	1	#	3	C	S	c	s	ຈ	ນ	ົ	້	໌		
0	1	0	0	\$	4	D	T	d	t	ດ	ຕ	ຖ	-	ີ	ເ	
0	1	0	1	%	5	E	U	e	u	ດ	ຕ	ວ	-	ີ	ເ	
0	1	1	0	&	6	F	V	f	v	ມ	ກ	ຖ	-	ີ	ເ	
0	1	1	1	'	7	G	W	g	w	ເ	ກ	າ	-	ດ	ເ	
1	0	0	0	(	8	H	X	h	x	ຈ	ນ	ຫ	-	ດ	ເ	
1	0	0	1	)	9	I	Y	i	y	ແ	ນ	ຫ	-	ດ	ເ	
1	0	1	0	*	:	J	Z	j	z	ຂ	ບ	ສ	-	ດ	ເ	
1	0	1	1	+	;	K	[	k	{	ຂ	ປ	ທ	-	ດ	ເ	
1	1	0	0	.	<	L	\	l		ກ	ມ	ນ	-	ດ	ເ	
1	1	0	1	-	=	M	]	m	}	ງ	ຝ	ອ				
1	1	1	0	-	>	N	^	n	~	ງ	ັ	ສ				
1	1	1	1	/	?	O	-	o		ງ	ັ	ູ	ບ			

หน่วยความจำ	
ນ	10111001
ນ	10111001
ຖ	10110111
ບ	10111010
ໆ	11011000
ຈ	11000011
ສ	11010101

## ตารางแสดงรหัส ASCII แทนตัวอักษรภาษาอังกฤษและภาษาไทย

b7	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1			
b6	0	0	0	0	1	1	1	0	0	0	0	1	1	1	1			
b5	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1			
b4	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0			
b3	b2	b1	b0															
0	0	0	0					0	@	P	'	p		ສ	ກ	ຂ	ໜ	ໝ
0	0	0	1					!	1	A	Q	a	q	ກ	ທ	ມ	ຂ	ໝ
0	0	1	0					"	2	B	R	b	r	ຂ	ຜ	ຍ	າ	ໄ
0	0	1	1					#	3	C	S	c	s	ໝ	ໝ	ໜ	ໍ	ໜ
0	1	0	0					\$	4	D	T	d	t	ມ	ຕ	ດ	ໜ	ໝ
0	1	0	1					%	5	E	U	e	u	ໜ	ໝ	ລ	ໜ	ໝ
0	1	1	0					&	6	F	V	f	v	ໝ	ດ	ກ	ໜ	ໝ
0	1	1	1					'	7	G	W	g	w	ເ	ກ	ວ	ໜ	ໝ
1	0	0	0					(	8	H	X	h	x	ບ	ຫ	ສ	ໜ	ໝ
1	0	0	1					)	9	I	Y	i	y	ຈ	ນ	ໝ	ໜ	ໝ
1	0	1	0					*	:	J	Z	j	z	ໜ	ບ	ສ	ໜ	ໝ
1	0	1	1					+	;	K	[	k	{	ໜ	ປ	ທ	ໜ	ໝ
1	1	0	0					,	<	L	\	l		ໝ	ພ	ໜ	ໜ	ໝ
1	1	0	1					-	=	M	]	m	}	໘	ຜ	ອ	ໜ	ໝ
1	1	1	0					.	>	N	^	n	~	ກ	ພ	ສ	ໜ	ໝ
1	1	1	1					/	?	O	-	o		ກ	ພ	ຊ	໘	ໝ



# เลขฐานกับรหัส ASCII

ค่าของ ‘A’ เมื่อถูก 나타งาหัส ASCII มีค่าเท่ากับ 01000001 ฐานสอง

ซึ่งเมื่อแปลงเป็นเลขฐานสิบ จะมีค่าเท่ากับ 65

ในการเขียนโปรแกรมสามารถแปลงชนิดของข้อมูลจากตัวเลขจำนวนเต็ม  
ไปเป็นตัวอักษรได้ ถ้าตัวเลขนั้นตรงกับรหัส ASCII

ดังนั้นถ้ามีการแปลงเลข 65 ไปเป็นตัวอักษร จะมีค่าเท่ากับ ‘A’

# ตารางเปรียบเทียบรหัส ASCII กับเลขฐานสิบ

<b>Code</b>	<b>Char</b>										
32	[space]	48	0	64	@	80	P	96	`	112	p
33	!	49	1	65	A	81	Q	97	a	113	q
34	"	50	2	66	B	82	R	98	b	114	r
35	#	51	3	67	C	83	S	99	c	115	s
36	\$	52	4	68	D	84	T	100	d	116	t
37	%	53	5	69	E	85	U	101	e	117	u
38	&	54	6	70	F	86	V	102	f	118	v
39	'	55	7	71	G	87	W	103	g	119	w
40	(	56	8	72	H	88	X	104	h	120	x
41	)	57	9	73	I	89	Y	105	i	121	y
42	*	58	:	74	J	90	Z	106	j	122	z
43	+	59	;	75	K	91	[	107	k	123	{
44	,	60	<	76	L	92	\	108	l	124	-
45	-	61	=	77	M	93	]	109	m	125	}
46	.	62	>	78	N	94	^	110	n	126	~
47	/	63	?	79	O	95	_	111	o	127	[backspace]

# ภาษาสำหรับนักพัฒนาโปรแกรม

- **วิชวลเบสิก (VISUAL Basic)**

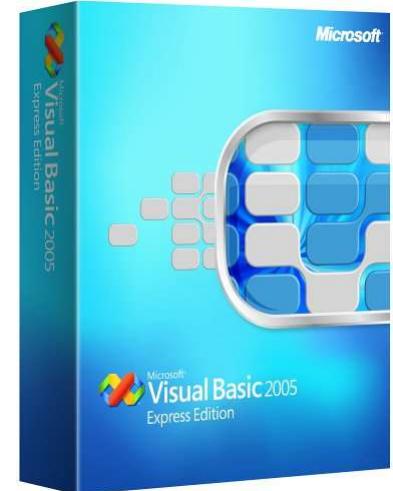
พัฒนาโดยบริษัทไมโครซอฟต์ ปัจจุบันใช้เทคโนโลยี .net

- **ภาษาจาวา (JAVA)**

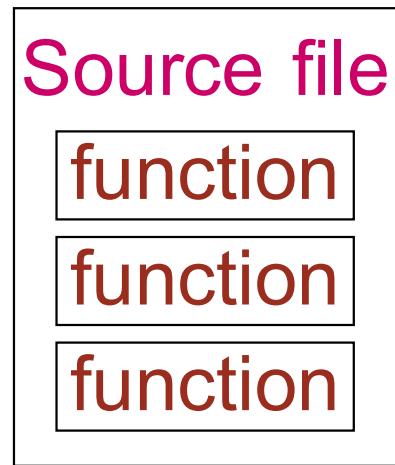
พัฒนาโดยบริษัท Sun Microsystem

- **ภาษาซีชาร์ฟ์ (C#)**

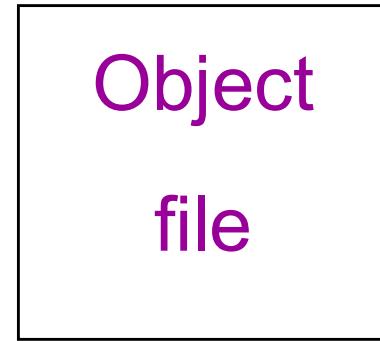
พัฒนาโดยไมโครซอฟต์ ปัจจุบันใช้เทคโนโลยี .net



# ขั้นตอนการสร้างโปรแกรมด้วยภาษา C



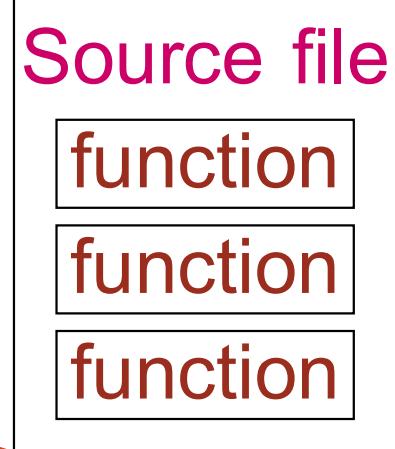
*compile*



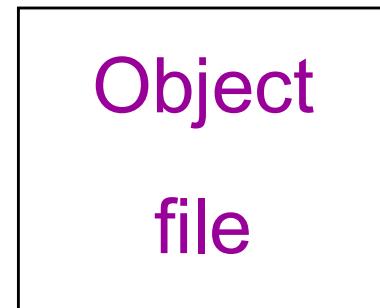
*link*



*link*



*compile*



*link*

↗

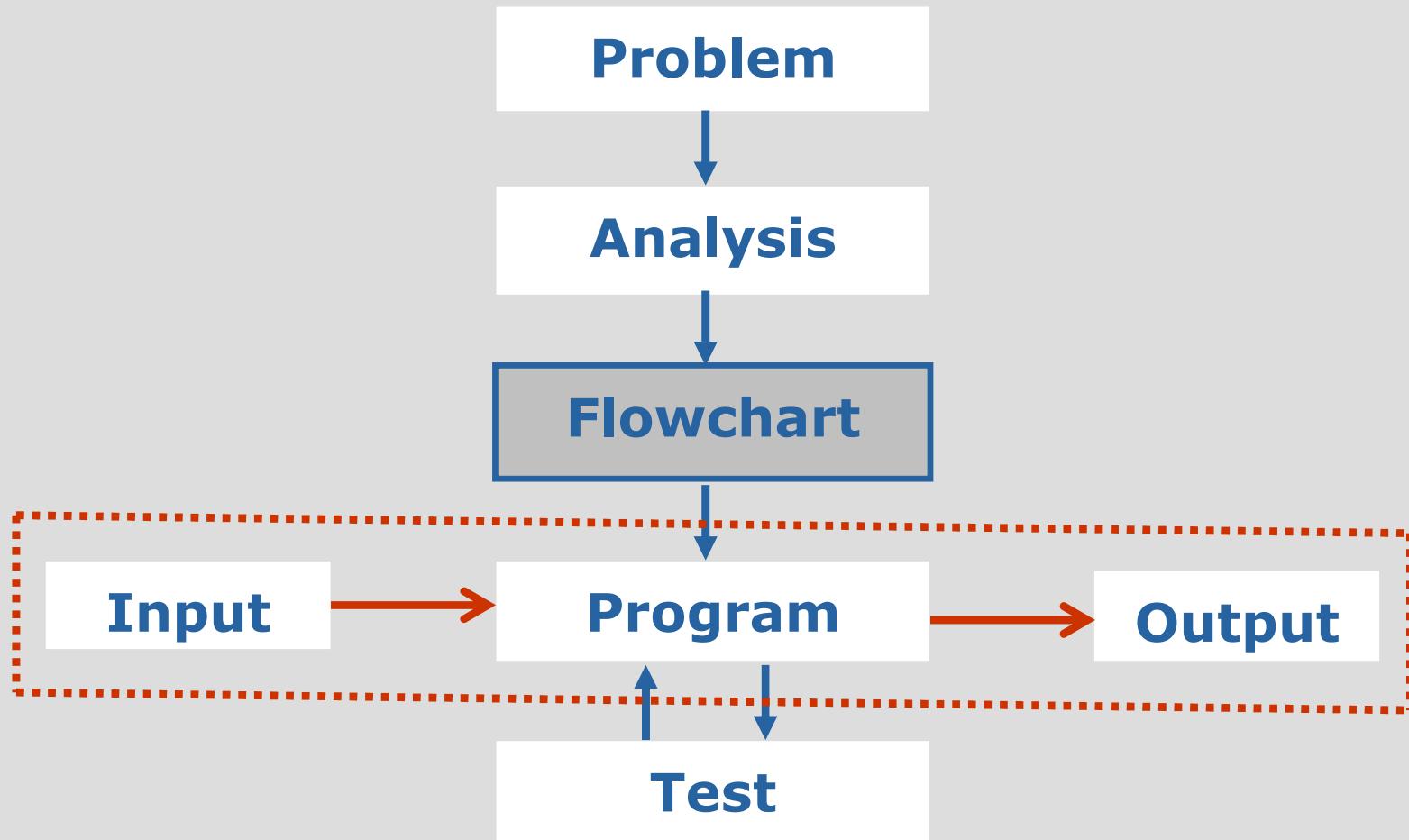
# 1.3 การพัฒนาโปรแกรม

## ● การพัฒนาโปรแกรม ประกอบด้วย 5 ขั้นตอน

1. กำหนดและวิเคราะห์ปัญหา (State problem & Problem analysis)
2. เขียนผังงาน (Flowchart)
3. เขียนโปรแกรม (Program Development)
4. ทดสอบและแก้ไขโปรแกรม (Testing & debugging)
5. ทำเอกสารและบำรุงรักษาโปรแกรม (Document & maintenance)

# ขั้นตอนพัฒนาโปรแกรม

- สรุป 5 ขั้นตอนในการพัฒนาโปรแกรม



# 1.4 กำหนด-วิเคราะห์ปัญหา

- กำหนดขอบเขตของปัญหา ให้ชัดเจนว่า จะให้คอมพิวเตอร์ทำอะไร (What?)
- วิเคราะห์ปัญหา (Problem analysis) กำหนด
- Input: ลักษณะของข้อมูลเข้า
- Process: วิธีการประมวลผล (How?)
- Output: ลักษณะของผลลัพธ์ที่ต้องการ

# ตัวอย่าง 1.1

- ออกแบบโปรแกรมให้คอมพิวเตอร์ทำงานเป็นเครื่องคิดเลขอย่างง่าย โดยรับข้อมูล 2 ค่า ( $X, Y$ ) และแสดงผลบวกทางจอภาพ

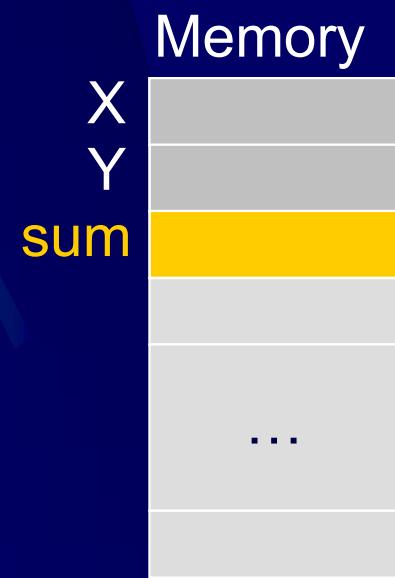
- **Problem:** คำนวณผลบวกของ 2 ค่า

- **Problem Analysis**

1. **Input:** รับข้อมูล ( $X, Y$ ) จากคีย์บอร์ด

2. **Process:** คำนวณ  $sum = X + Y$

3. **Output:** แสดงผลบวก ( $sum$ ) ทางจอภาพ



# ตัวอย่าง 1.2

- ออกแบบโปรแกรมให้คอมพิวเตอร์รับข้อมูล 3 ค่า  
คำนวณค่าเฉลี่ย และแสดงค่าเฉลี่ย ทางจอภาพ

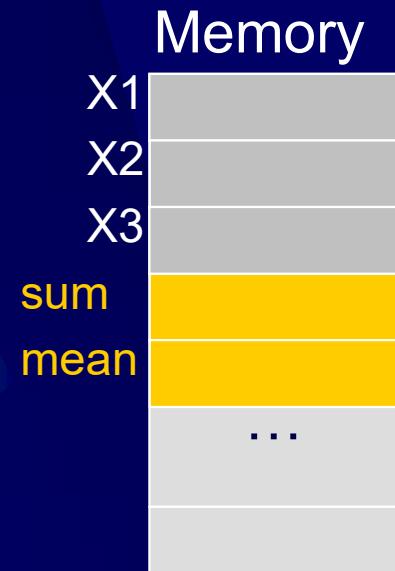
- **Problem:** คำนวณค่าเฉลี่ยของ 3 ค่า  $((X_1 + X_2 + X_3) / 3)$

## ● Problem Analysis

1. Input: รับข้อมูล ( $X_1, X_2, X_3$ )
2. Process: คำนวณ  $sum = X_1 + X_2 + X_3$

$$mean = sum / 3$$

3. Output: แสดงค่าเฉลี่ย (mean) ทางจอภาพ



# ตัวอย่าง 1.3

- ออกแบบโปรแกรมให้คอมพิวเตอร์รับข้อมูล N ค่า  
คำนวณค่าเฉลี่ย และแสดงค่าเฉลี่ย ทางจอภาพ

- **Problem:** คำนวณค่าเฉลี่ยของ N ค่า ( $\sum_i^N X_i / N$ )

## ● Problem Analysis

1. Input: รับข้อมูล N และ X ( $X_1, \dots, X_N$ )

2. Process:

loop for (i=1 to N)

    read X

    sum = sum + X

end for

mean = sum/N

3. Output: แสดงค่าเฉลี่ย (mean) ทางจอภาพ



# 1.5 การเขียนผังงาน

● **ผังงาน (Flowchart)** เป็นแผนภาพ ที่ใช้อธิบาย  
ลำดับการทำงานของโปรแกรมเป็นขั้นตอน

● **ประโยชน์ของ Flowchart**

1. อธิบายลำดับขั้นตอนการทำงานของโปรแกรม
2. ทำให้ตรวจสอบข้อผิดพลาดของโปรแกรมได้ง่าย
3. ทำให้ผู้อื่นสามารถศึกษาการทำงานและแก้ไข
  - โปรแกรมได้ง่าย

# 1.5.1 ประเภทของผังงาน

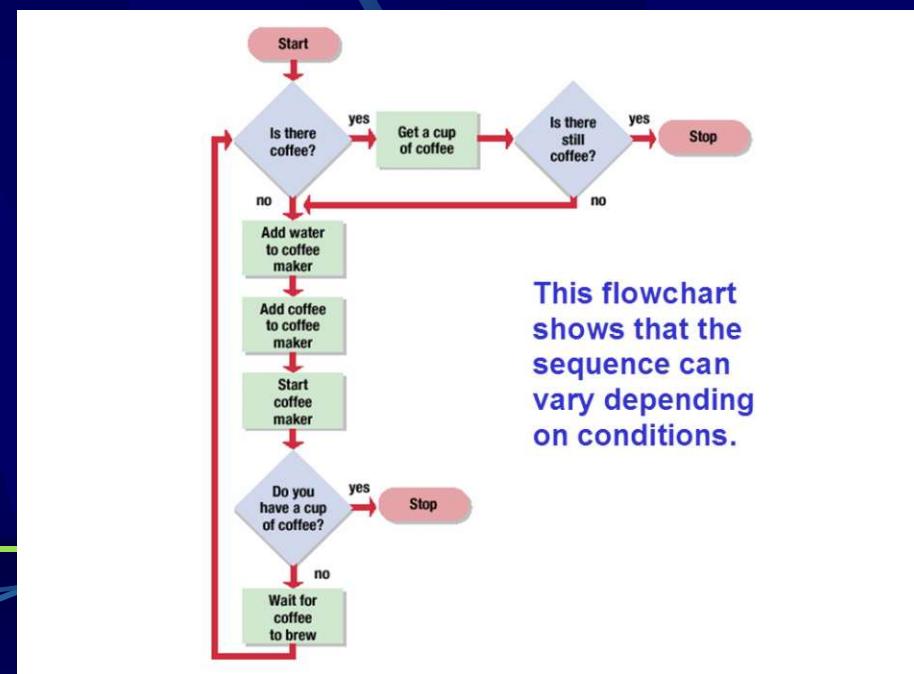
## ผังงานระบบ (System Flowchart)

แสดงขั้นตอนการทำงานภายในระบบงาน  
โดยจะกล่าวอย่างกว้างๆ



## ผังงานโปรแกรม (Program Flowchart)

แสดงขั้นตอนของคำสั่ง  
ที่ใช้ในโปรแกรม



## 1.5.2 สัญลักษณ์ในผังงาน

### สัญลักษณ์พื้นฐาน ที่นิยมใช้ใน Flowchart

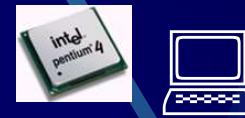
การเริ่มต้น และการสิ้นสุดการทำงาน



ลูกศรแสดงทิศทางการทำงาน และการไหลของข้อมูล



การประมวลผล หรือการคำนวณ



การรับข้อมูล หรือแสดงผล (ไม่ระบุชนิดอุปกรณ์)



การแสดงผลทางจอภาพ



การแสดงผลทางเครื่องพิมพ์ (เป็นเอกสาร)



การตรวจสอบเงื่อนไข (เพื่อการตัดสินใจเมื่อมีทางเลือก)



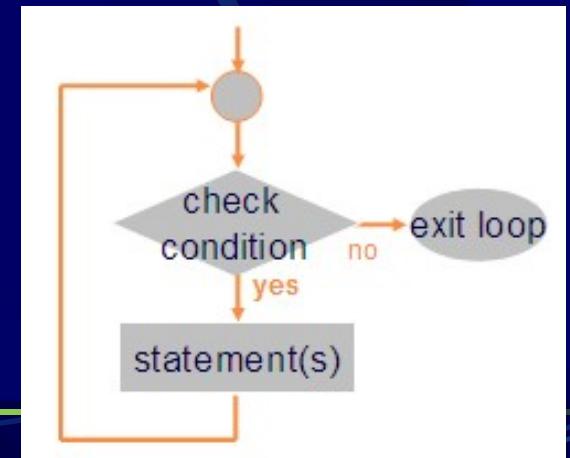
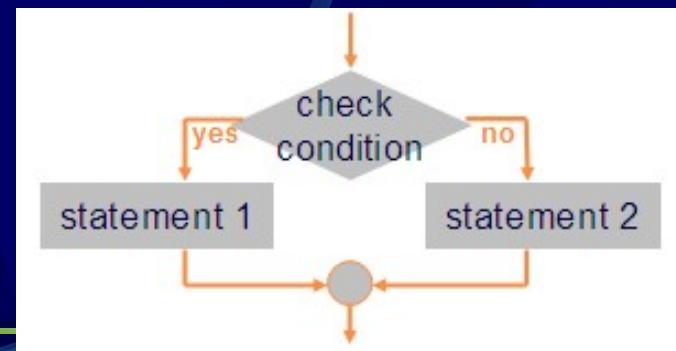
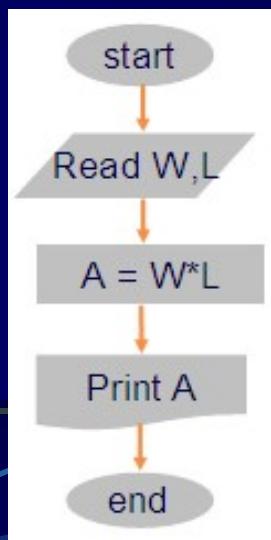
จุดเชื่อมต่อของผังงาน

## 1.5.3 รูปแบบของผังงาน

แบบลำดับ (Sequence)

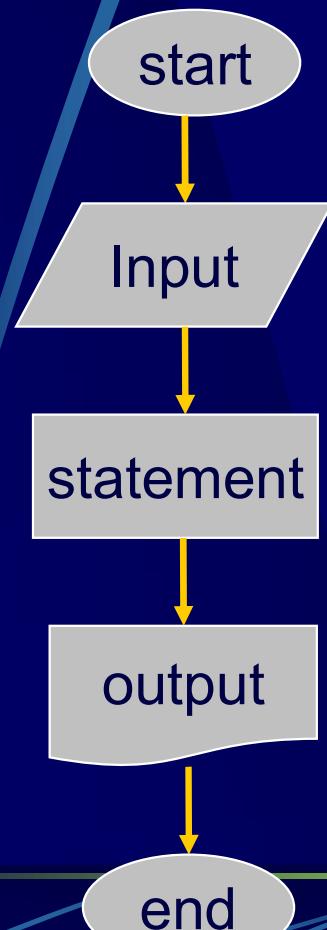
แบบมีทางเลือก (Selection)

แบบทำซ้ำ (Looping)



### 1.5.3.1 ผังงานแบบลำดับ

#### Flowchart แบบลำดับ (Sequence)

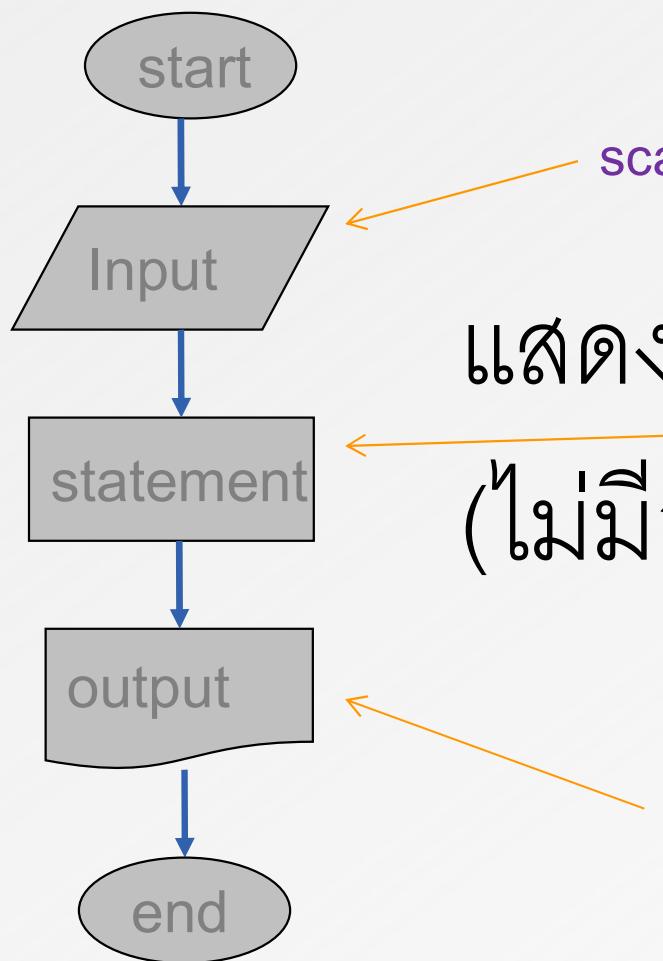


แสดงขั้นตอนการทำงานที่เรียงลำดับ  
(ไม่มีการข้ามขั้น หรือย้อนกลับ)

# ผังงานแบบลำดับ



## • Flowchart แบบลำดับ (Sequence)

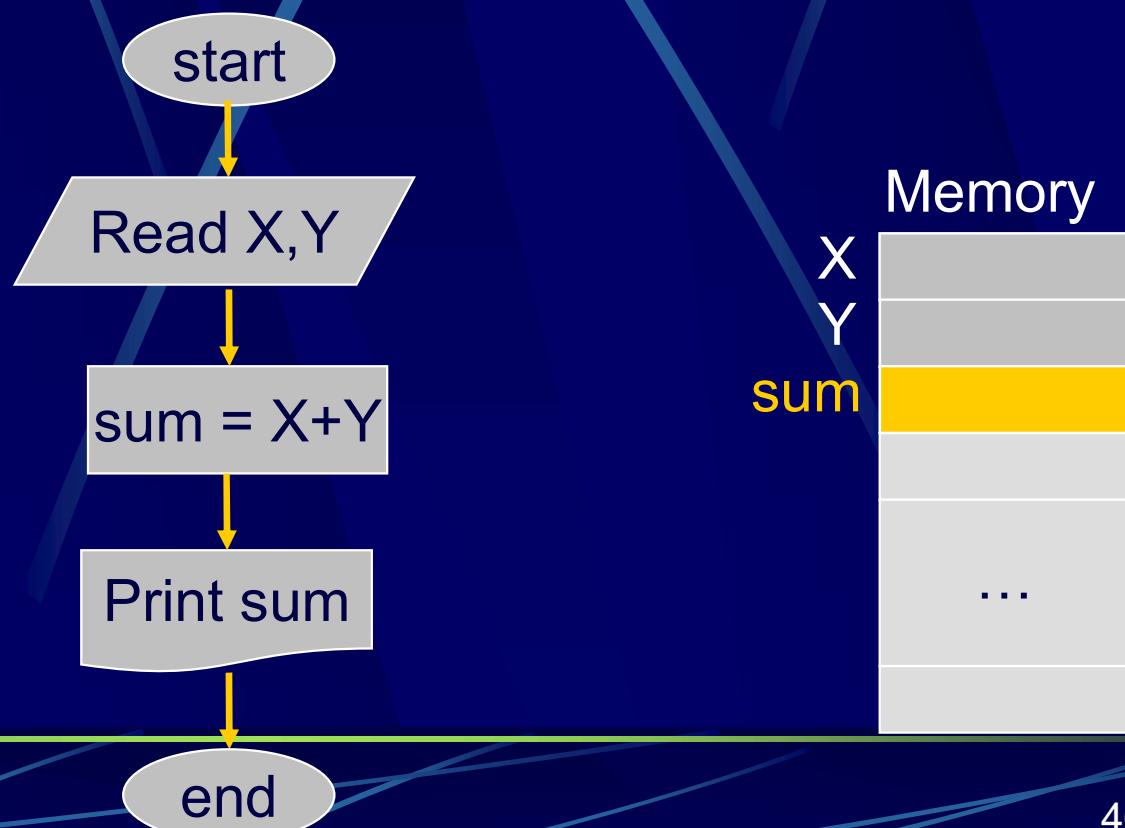


```
#include "stdio.h"
#include "conio.h"
main()
{
    int x,y;
    scanf("%d",&x);
    y = x + 1;
    printf("Output = %d\n",y);
}
```

getch();

# ตัวอย่าง 1.4

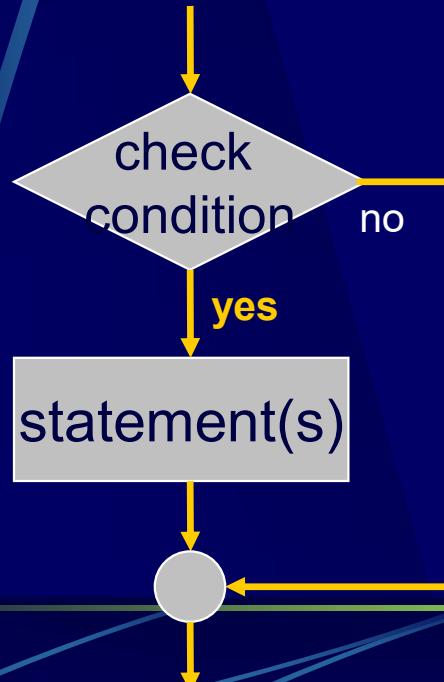
แสดงการออกแบบ Flowchart เพื่อให้โปรแกรมทำงานเป็นเครื่องคิดเลขอย่างง่าย โดยรับข้อมูล 2 ค่า ( $X, Y$ ) คำนวณ การบวก (sum) พิจารณาแสดงผลบวก



## 1.5.3.2 ผังงานแบบมีทางเลือก

- แสดงการตรวจสอบเงื่อนไขให้โปรแกรมเลือกทำอย่างใดอย่างหนึ่ง ซึ่งมี 3 กรณี

### 1. การเลือกแบบ 1 เส้นทาง



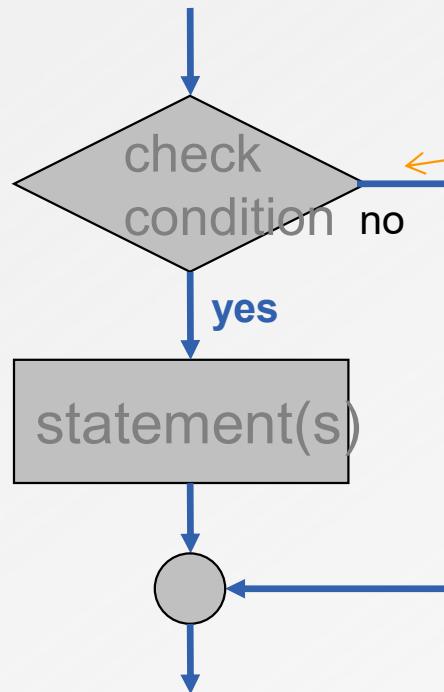
จะทำงานเฉพาะเมื่อเงื่อนไข  
เป็นจริงเท่านั้น

# ผังงานแบบมีทางเลือก



- แสดงการตรวจสอบเงื่อนไขให้โปรแกรมเลือกทำอย่างใดอย่างหนึ่ง ซึ่งมี 3 กรณี

## 1. การเลือกแบบ 1 เส้นทาง



คำสั่ง **if** ไม่ต้องมี **else**

จะทำงานเฉพาะเมื่อเงื่อนไข  
เป็นจริงเท่านั้น

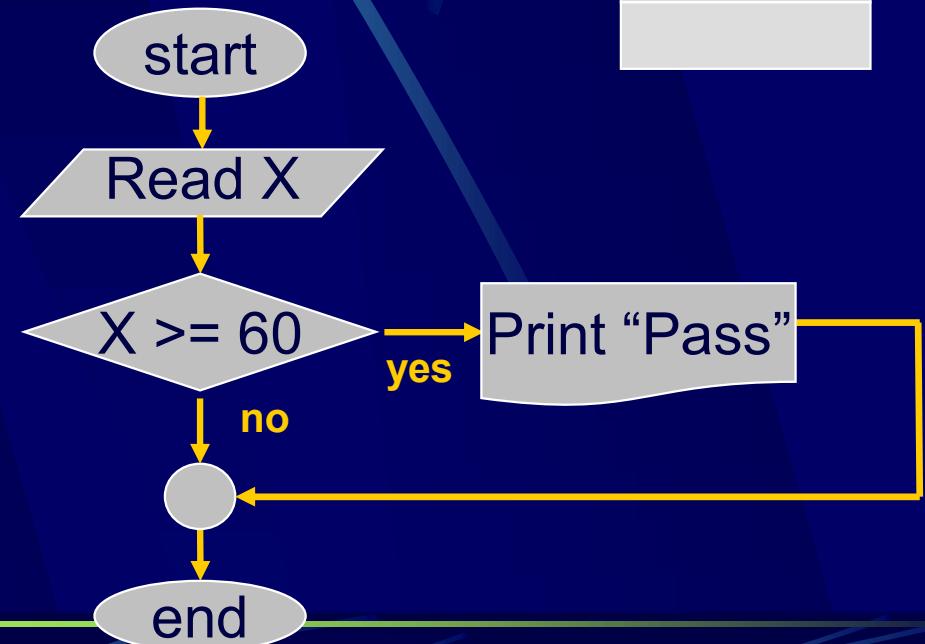
# ตัวอย่าง 1.5

แสดงการออกแบบ **Flowchart** เพื่อให้  
โปรแกรมรับคะแนนนักศึกษา (X)  
ตรวจสอบและแสดงผล ถ้าผ่าน (Pass)  
โดยมีเงื่อนไขดังนี้

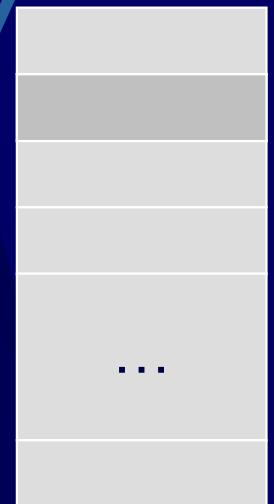
เงื่อนไข

คะแนน  $\geq 60$  ผ่าน (Pass)

คะแนน  $< 60$  ตก (Fail)

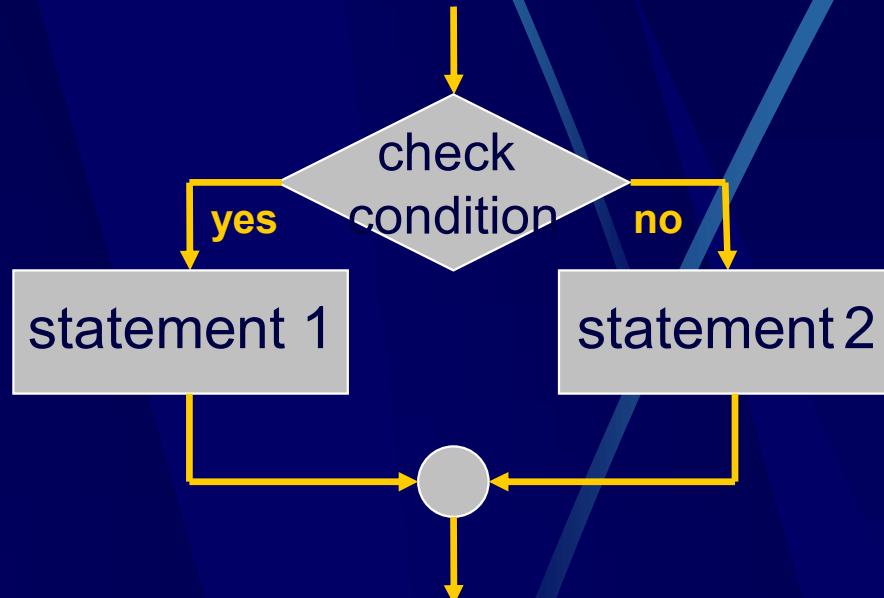


Memory



# ผังงาน-ทางเลือก

## 2. การเลือกแบบ 2 เส้นทาง

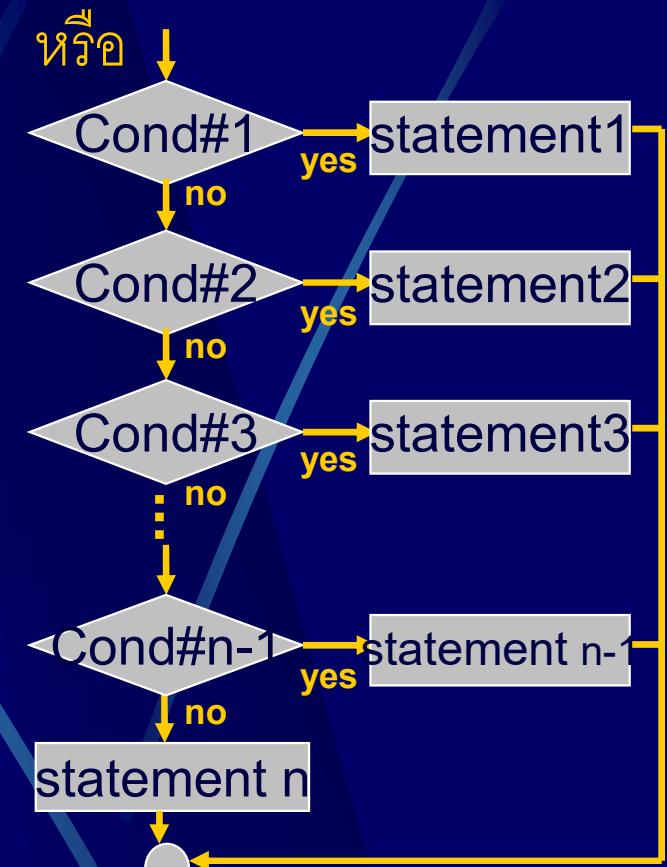
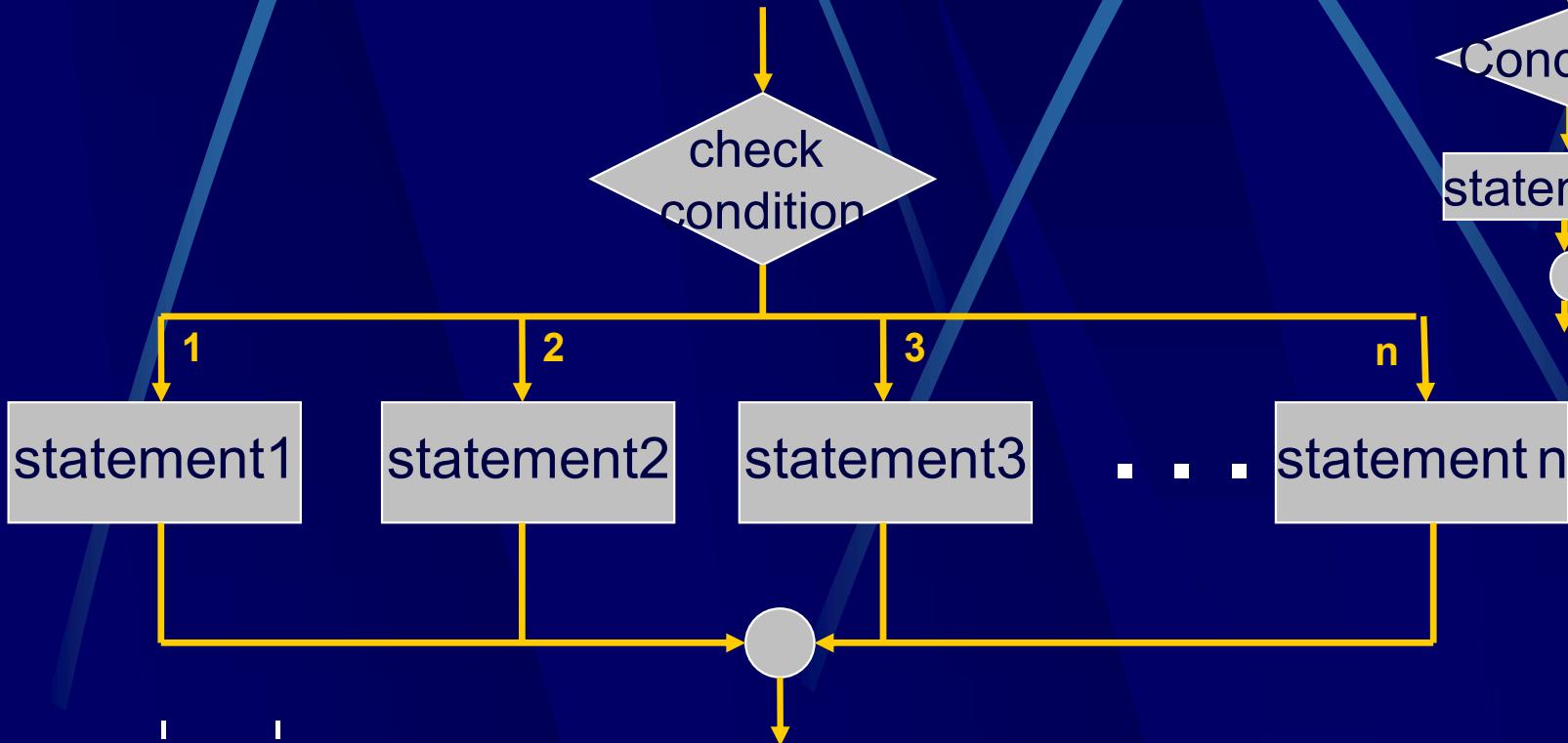


เมื่อเงื่อนไขเป็นจริงจะทำอย่างหนึ่ง

เมื่อเงื่อนไขเป็นเท็จจะทำอย่างหนึ่ง

# ผังงาน-ทางเลือก

## 3. การเลือกแบบหลายเส้นทาง (n)



เมื่อเงื่อนไขเท่ากับทางเลือกใดจะทำตามทางนั้น

# ตัวอย่าง 1.6

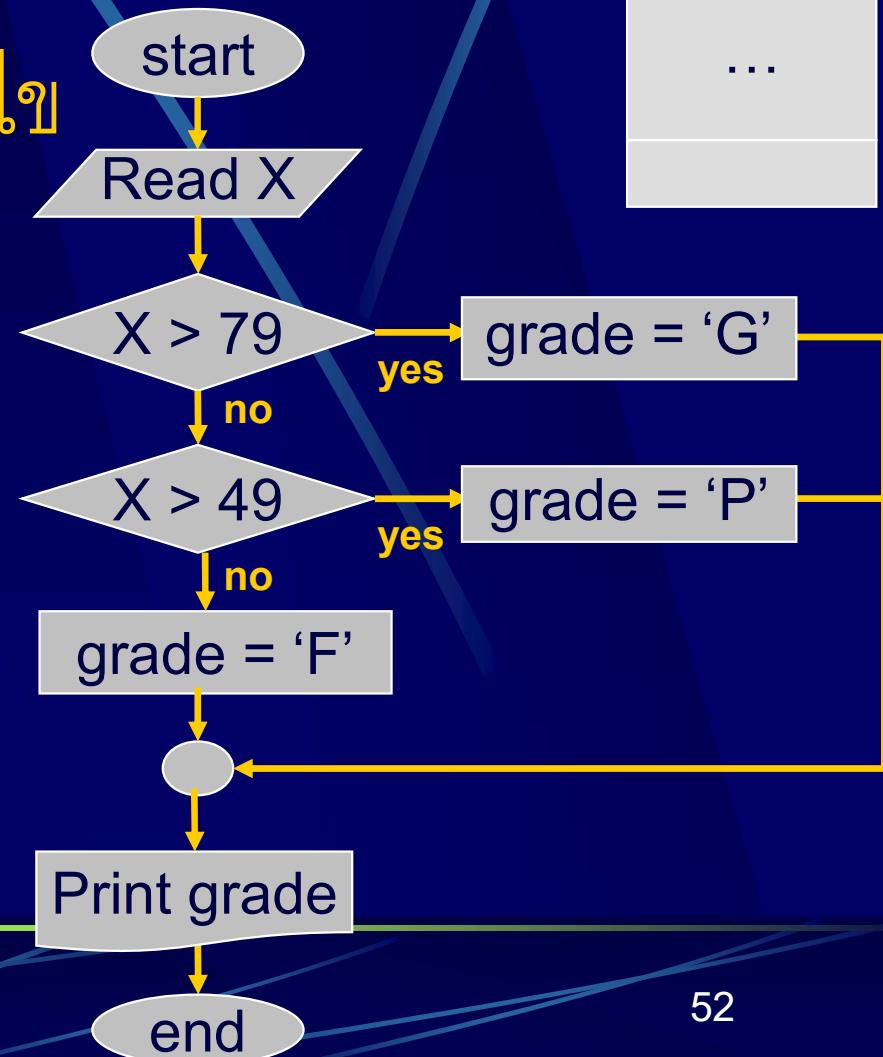
แสดงการออกแบบ Flowchart เพื่อให้โปรแกรมรับคะแนนนักศึกษา (X) ตรวจสอบและจัดกลุ่มตามเงื่อนไข พร้อมแสดงผลลัพธ์

เงื่อนไข

คะแนน 80-100 กลุ่ม G

คะแนน 50-79 กลุ่ม P

คะแนนต่ำกว่า 50 กลุ่ม F



Memory

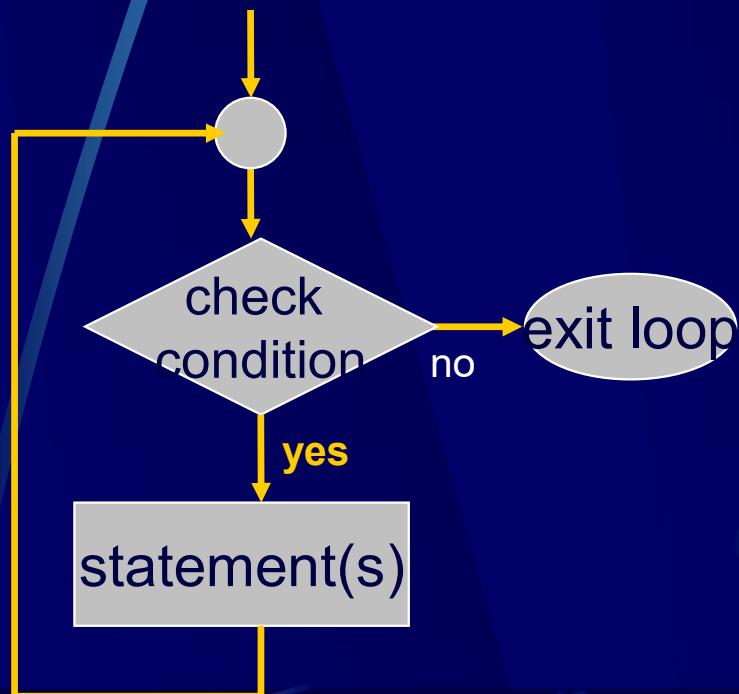
X  
grade

...

### 1.5.3.3 ผังงานแบบทำซ้ำ

- การทำซ้ำ (Looping) แบ่งเป็น 3 กรณี

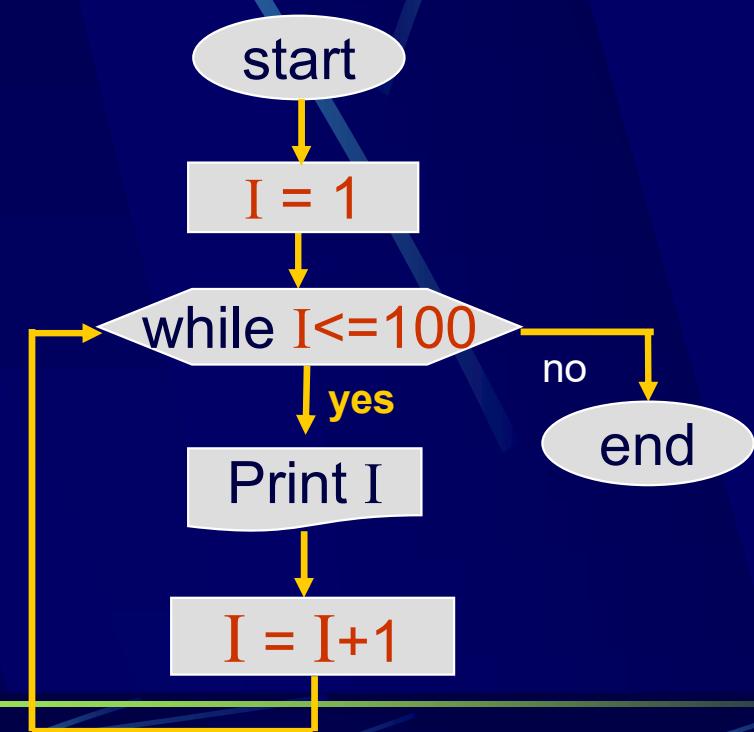
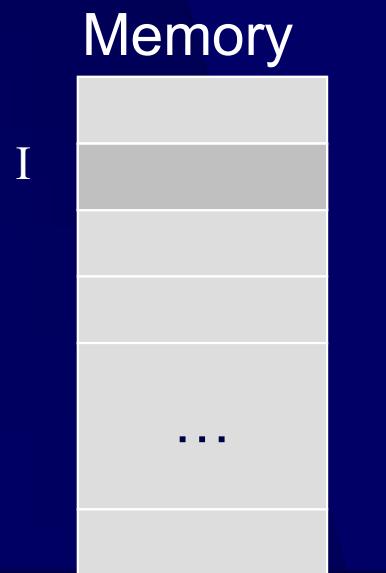
- 1. การทำซ้ำเมื่อเงื่อนไขเป็นจริง



ตรวจสอบเงื่อนไขก่อน  
จะทำงาน (Statement) ซ้ำ  
เมื่อเงื่อนไขเป็นจริง  
(ออกจากทำงานซ้ำเมื่อเงื่อนไขเป็นเท็จ)

# ตัวอย่าง 1.7

- แสดงการออกแบบ Flowchart เพื่อให้โปรแกรมคำนวณการกำหนดค่าที่เพิ่มทีละ 1 จาก 1 – 100
- กำหนดให้  $I = 1, 2, 3, \dots, 100$

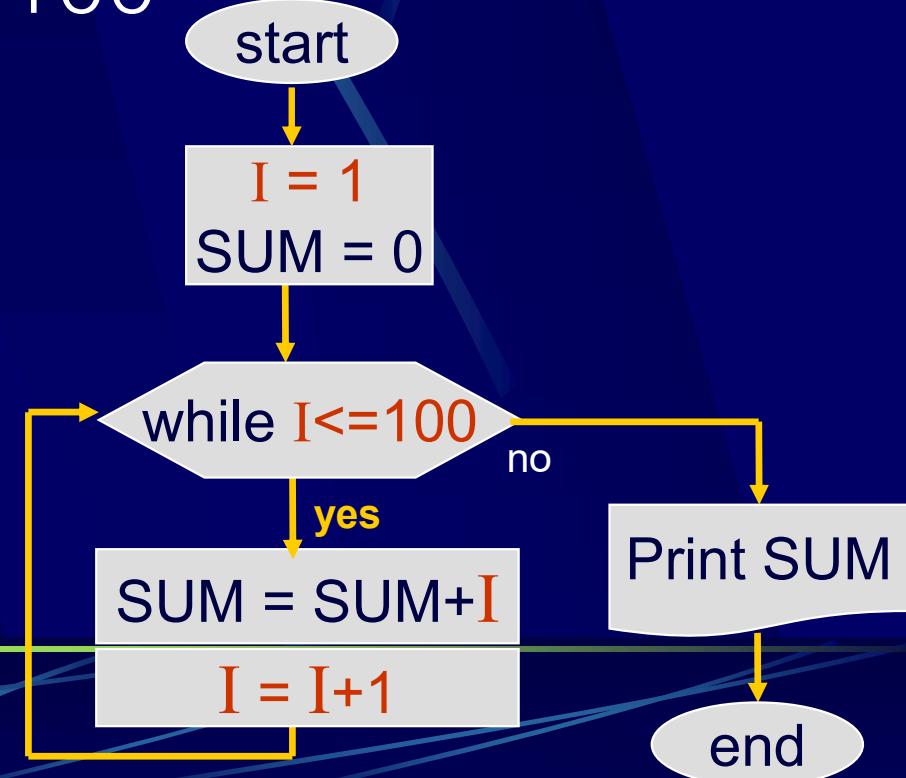
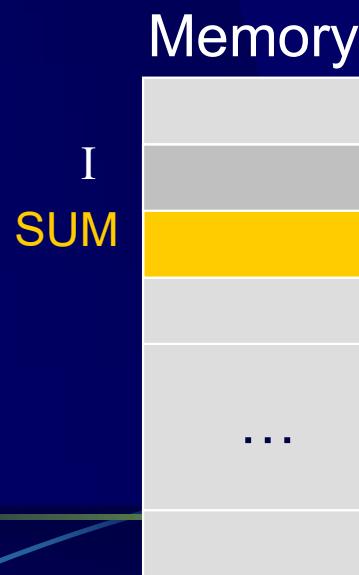


# ตัวอย่าง 1.8

- แสดงการออกแบบ Flowchart เพื่อให้โปรแกรมคำนวณ การบวก  $1+2+3+\dots+100$

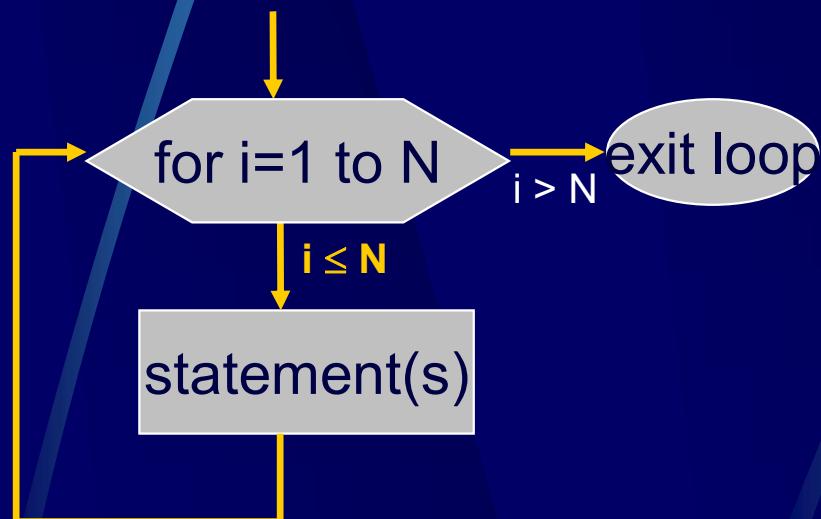
- กำหนดให้  $I = \underbrace{1, 2, 3, \dots, 100}$

และ  $SUM = 1+2+3+\dots+100$



# ผังงาน-ทำซ้ำ

## 2. การทำซ้ำตามจำนวนที่ระบุ

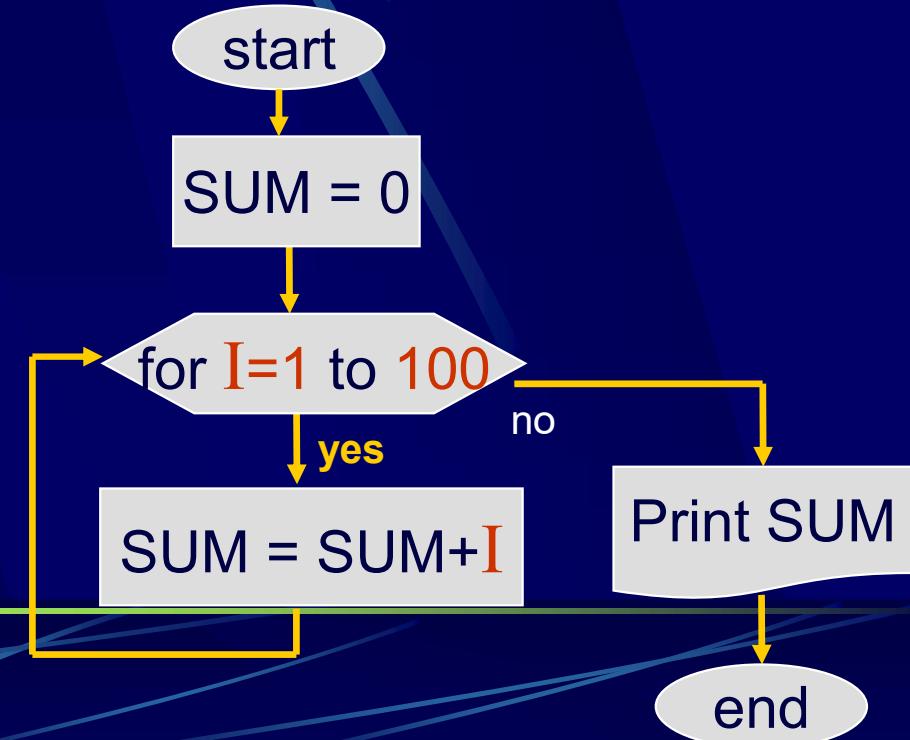
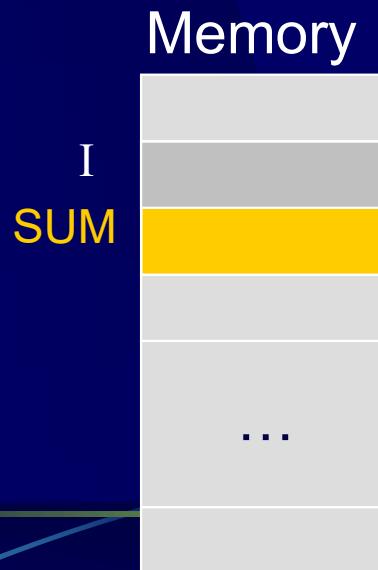


ทำงานตามรอบที่กำหนด  
โดยเริ่มจากการ**รอบเริ่มต้น** ( $i=1$ )  
**ไปยังรอบสุดท้าย** ( $i=N$ )  
(ปกติการนับรอบจะเพิ่มที  
ละ 1 ครั้ง ( $i = i+1$ ))

# ตัวอย่าง 1.9

- แสดงการออกแบบ Flowchart เพื่อให้โปรแกรมคำนวณ การบวก  $1+2+3+\dots+100$
- กำหนดให้  $I = \underbrace{1, 2, 3, \dots, 100}$

และ  $SUM = 1+2+3+\dots+100$

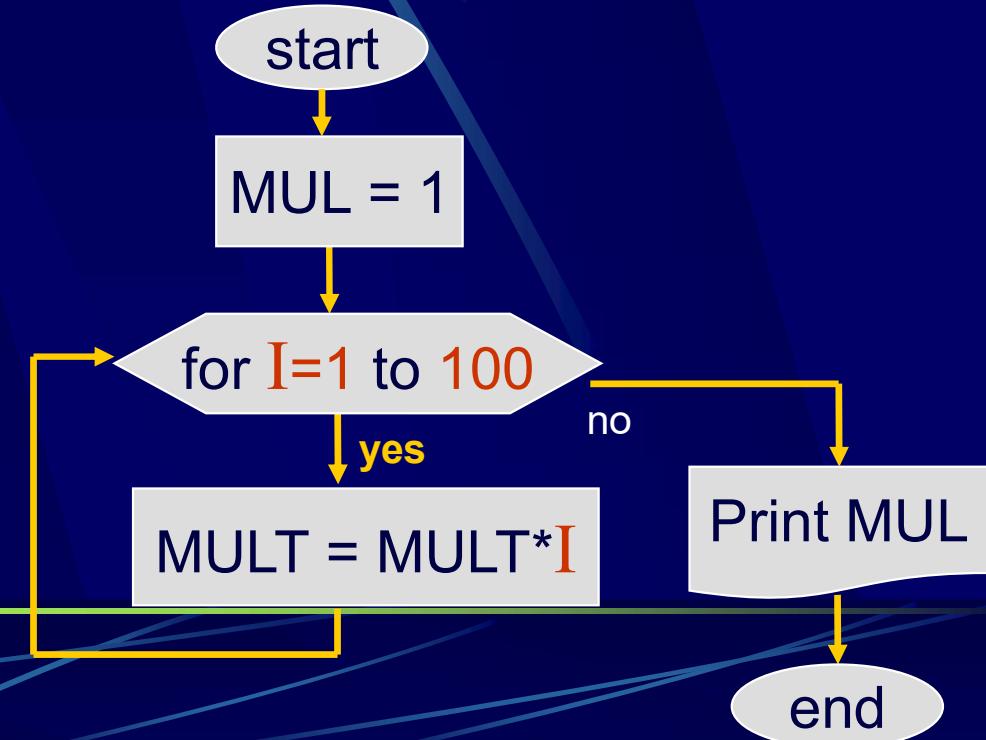
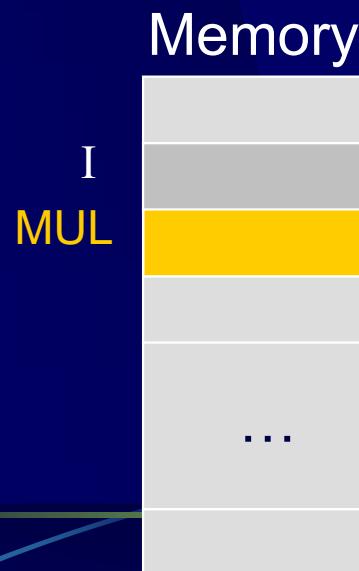


# ตัวอย่าง 1.10

- แสดงการออกแบบ Flowchart เพื่อให้โปรแกรมคำนวณ การคูณ  $1 \times 2 \times \dots \times 100$

- กำหนดให้  $I = 1, 2, 3, \dots, 100$

และ  $MUL = 1 \times 2 \times 3 \times \dots \times 100$



## ข้อคำนึง

- การเขียนผังงานหรือ flowchart ต้องไม่ยึดติดกับภาษา ผังงานที่เขียนต้องสามารถนำไปใช้เขียนโปรแกรมได้ทุกภาษา