ASM Language	**ใบงานที่ 1** NASM Basic Instructions และ Addressing Modes
	สัปดาห์ที่ 1 **ระยะเวลา** ในชั้นเรียน **01418233 แอสแซมบลีและสถาปัตยกรรมคอมพิวเตอร์** ชื่อ นายบุญทกานต์ ศิริกมลศิพงษ์ รหัสนิสิต 6430200418

เวลา : 2 ชั่วโมง

จุดประสงค์การเรียนรู้

1. ทำการเรียนรู้การเขียนโปรแกรมภาษา Assembly
2. ทำการเรียนรู้การกำหนดชนิดข้อมูลและ Symbolic

เนื้อหาการเรียนรู้

1. Data Transfer Instructions

2. Addition และ Subtraction

3. Data Addressing Modes

เนื้อหา

1. Assembly – Syntax เบื้องต้น

Assembly program สามารถแบ่งการเขียนโปรแกรมได้ 3 sections ดังนี้

1.1 Data section เป็นส่วนที่ถูกใช้ประกาศข้อมูลเริ่มต้นหรือกำหนดค่าคงที่ สามารถกำหนดชื่อ ระดับของข้อมูล ขนาดของข้อมูล เป็น

รูปแบบ (syntax) สำหรับ data section คือ

section.data

1.2 **Bss** section คือ ส่วนประกาศตัวแปร

รูปแบบ (syntax) สำหรับ bss section คือ

section.bss

1.3 **Text** section คือ ส่วนนำคำสั่งที่ใช้สำหรับเขียนโปรแกรม ในส่วนนี้เริ่มต้นต้องประกาศ **global _start**, ซึ่งเป็นการบอกระบบ (kernel) จุดเริ่มต้นของการ execution โปรแกรม

รูปแบบ (syntax) สำหรับ text section คือ

section.text

```
global _start

_start:
```

การ **Compiling** และ **Linking** กับโปรแกรมภาษา **Assembly** ใน **NASM**

nasm **และ** ld **ถูกติดตั้งในระบบและถูกเซ็ตตัวแปรแวดล้อม** PATH **เรียบร้อย ซึ่งขั้นตอน**สำหรับ compiling **และ** linking **ได้ดังนี้**

เขียนโปรแกรมจาก text editor **และ** save **ตัวอย่าง** hello.asm

ทำการ assemble **โปรแกรมที่เขียนด้วย**

```
nasm -f elf64 -o hello.o hello.asm
```

ถ้าหากมีการผิดพลาดของโปรแกรมจะแสดงและกลับไปแก้ไขให้ถูกต้อง หรือ หากไม่มีข้อผิดพลาด จะได้ไฟล์ชื่อ hello.o

ทำการ link กับ object file **และสร้างไฟล์** executable **ชื่อ** hello**ด้วยคำสั่ง**

```
ld  -o hello hello.o
```

ทำการ Execute **โปรแกรมด้วยการพิมพ์**

```
./hello
```

โปรแกรม hello.asm

```
global _start
section .text
_start:
    mov     rax, 1 ; write
    mov     rdi, 1 ; stdout
    mov     rsi, msg
    mov     rdx, len
    syscall

    mov     rax, 60 ; exit
    mov     rdi, 0
    syscall

section .data
msg:    db      "Hello, world!", 10
.len:   equ     $ - msg
```

2. Data Transfer Instructions

Data transfer instructions คือการย้ายข้อมูลระหว่าง registers หรือ ระหว่าง registers และ memory. รูปแบบของคำสั่งนี้ดูได้จากด้านล่างเป็นการอธิบายแบบย่อหากต้องการดูคำอธิบายที่มากกว่านี้หาอ่านจาก lecture notes หรือ textbook.

MOV destination, source โยกย้ายข้อมูลจาก *source* ไปเก็บที่ *destination*.
MOVZX destination, source โยกย้ายข้อมูลจาก source ไปเก็บที่ destination พร้อม zero extension
MOVSX destination, source โยกย้ายข้อมูลจาก source ไปเก็บที่ destination พร้อม sign extension

ให้เขียนโปรแกรม moves.asm ด้านล่างนี้ เพื่อนำไปประกอบกับการฝึกปฏิบัติในหัวข้อทัดไป

```
; Demonstration of MOV, MOVZX, and MOVSX
        global _start
section .data
var1    dw 1000h
var2    dw 2000h

section .text
_start:
;   Demonstrating MOV and MOVZX
        mov     ax, 0A69Bh
        movzx   bx, al
        movzx   ecx,ah
        movzx   edx,ax

;   Demonstrating MOVSX
        movsx   bx, al
        movsx   ecx,ah
        movsx   edx,ax

;   Exit
        mov rax, 60
        mov rdi, 0
        syscall
```

1.1 ให้ทำการ Assemble and Link moves.asm

เปิดไฟล์หรือเขียนตามด้านบน *moves.asm* และ assemble กับ link เข้ากับ file. โดยใช้ **nasm และ ld** จากตัวอย่างข้อที่ 1

1.2 ให้ทำการ Debugger เพื่อหาค่าที่ถูกประมวลผลจากโปรแกรม moves

เปิดโปรแกรม gdb เพื่อ trace ไฟล์ execution
gdb ./moves
break _start
run
set disassembly-flavor intel

disassemble start

info registers

info all-register

info registers eax

x/2xb $variable

print/x $esp

print/x $eax

nexti

quit

สังเกตุการเปลี่ยนแปลงของ registers และ variables. Make the necessary corrections to your answers.

MOV and MOVZX

1) al, ah (hex) = 0x9b, 0xab	2) bx (hex) = 0x9b
3) ecx (hex) = 0x00ab	4) edx (hex) = 0x009b

MOVSX

5) bx (hex) = 0xff9b	
6) ecx (hex) = 0xffffff9b	7) edx (hex) = 0xffff9b

ตัวแปรมีค่าเท่าไร

var1 (hex) = 0x10
var2 (hex) = 0x20