Assembly Midterm

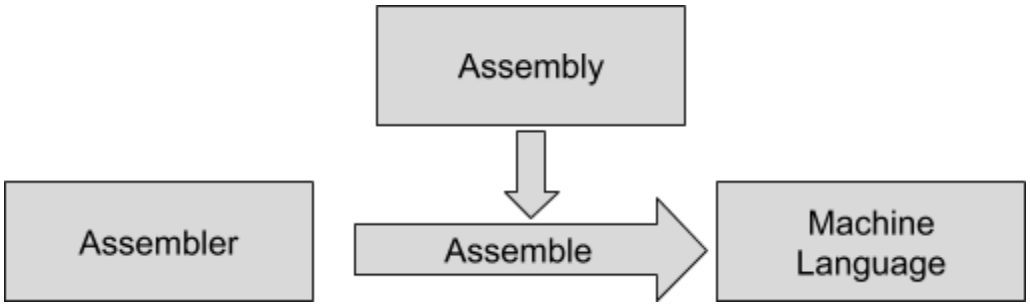
By Natthaphach Anuwattananon

Content

By Natthaphach Anuwattananon

ภาษา Assembler

- ภาษา Assembly จะถูก Assembler ทำการ Assemble กลายเป็น Machine Language



Register

- ใช้เก็บข้อมูล มีทั้งหมด 14 register แต่ล่ะอันมี 16 bit
- data และ address register
    - data register
        - AX แบ่งเป็น AH AL
        - BX แบ่งเป็น BH BL
        - CX แบ่งเป็น CH CL
        - DX แบ่งเป็น DH DL
    - pointer และ index register
        - SP (stack pointer)
        - BP (base pointer)
        - SI (source pointer)
        - DI (destination index)
    - segment register
        - CS (code segment)
        - DS (data segment)
        - SS (stack segment)
        - ES (extra segment)
- pointer และ index register
    - IP (index pointer)
- status flag register
    - CF bit 0 carry flag
    - PF bit 2 parity flag
    - AF bit 4 auxiliary flag
    - ZF bit 6 zero flag
    - IF bit 9 interrupt enable flag
    - DF bit 10 direction flag
    - OF bit 11 overflow flag

By Natthaphach Anuwattananon

Segment

- พื้นที่ที่ใช้ในการเก็บข้อมูล สามารถเก็บได้ถึง 64Kbyte
- แบ่งออกเป็น 4 ชนิด
    - CS code segment ใช้ในการเก็บ code program
    - DS data segment ใช้ในการเก็บข้อมูลที่ใช้
    - SS stack segment ใช้ในการเก็บข้อมูล, address ชั่วคราว
    - ES extra segment เก็บข้อมูลที่ใช้กับคำสั่ง string

Constants

- number constants
    - binary ลงท้ายด้วย B จำนวนลบทำ two's complement
    - decimal ลงท้ายด้วย D (ไม่มีก็ได้) จำนวนลบใส่เครื่องหมาย -
    - hexadecimal ลงท้ายด้วย H จำนวนลบทำ two's complement
      ขึ้นต้นด้วยตัวเลขเท่านั้น เช่น 0F3H

Source Statement

- Assembly language คำสั่ง processor ทำงานตอน execute
- Assembly directive คำสั่ง assembler ทำงานตอน assemble

Assembly-language instructions

[label:] mnemonic [operand] [;comment]

- label
    - ใช้อ้างอิง instruction
    - ประกอบด้วย A-Z 0-9 ? . @ _ $(ตัวเล็กไม่น่าได้ ไม่เห็นมีเขียน)
    - มี . ได้ แต่ต้องให้ตัวเลขเป็น .
- mnemonic
    - เป็นคำสั่งของ instruction
- operand
    - มี 0 ตัว
    - มี 1 ตัว <destination>
    - มี 2 ตัว <destination>,<source>
- comment
    - อย่าลืม ;

Assembler directives

- ใช้เพื่อกำหนด segment, define symbols, จองพื้นที่
- ไม่ถูก generate เป็น object code

Symbolic directive

[name] { **DB** | **DW** | **DD** } expression [, ...]

- เรียก name ว่า symbolic reference

DB	define byte	8 bit
DW	define word	16 bit
DD	define doubleword	32 bit

By Natthaphach Anuwattananon

DUP

n **DUP** <expression>

- ใช้ซ้ำค่าของ expression เป็นจำนวน n ตัว

DB 4 DUP (0) = DB 4 0, 0, 0, 0

?

- ใช้เพื่อจองเนื้อที่แบบไม่ระบุค่า

COUNT DB ?

Segment directive

SEGMENT - ENDS

<seg-name> **SEGMENT** [align-type] [combine-type] [‘class’]

…

<seg-name> **ENDS**

- seg-name
  - ชื่อของ segment ใช้ในคำสั่ง ASSUME
- align-type
  - PARA default address multiple 16
  - PAGE address multiple 256
- combine-type ใช้เพื่อระบุว่าถ้าเจอ segment ชื่อเหมือนกันต้องทำไง
  - ไม่ระบุ ไม่รวมกับ segment อื่น
  - PUBLIC segment ชื่อเดียวกัน มาต่อกัน
  - COMMON segment ชื่อเดียวกัน ใช้พื้นที่ร่วมกัน
  - STACK segment ที่จะใช้เป็น stack segment
- class
  - ใช้เพื่อจัดกลุ่ม segment ถ้ามี class เดียวกัน จะเก็บเรียงต่อกัน

ASSUME

ASSUME <seg-reg>:<seg-name> [, …]

- ใช้เพื่อเอา segment ที่สร้างขึ้นมา ใส่ลงไปใน segment register
- assume ต้องอยู่ใน code segment ต่อจาก SEGMENT directive

Procedure Directive

PROC - ENDP

<proc-name> **PROC** [**NEAR** | **FAR**]

…

<proc-name> **ENDP**

- ใช้ในการสร้าง procedure
- distance attribute
  - NEAR default เรียกได้จาก code segment ที่สร้าง procedure นี้
    เรียกได้จาก code segment ใน module อื่น ที่ชื่อเหมือนกัน
  - FAR เรียกได้จาก code segment อื่น

CALL

CALL <proc-name>

- ใช้ในการเรียกใช้ procedure

By Natthaphach Anuwattananon

END

END <proc-name>

- ใช้ระบุ procedure ที่จะเริ่มต้นทำงาน
- อยู่นอก code segment

Listing directive

PAGE

PAGE [lines] [, columns]

- ใช้เพื่อกำหนดขนาดของแต่ละหน้าของ listing

lines	default 57	range 10 - 255	กำหนดจำนวนบรรทัด
columns	default 80	range 60 - 132	กำหนดจำนวนตัวหนังสือต่อบรรทัด

TITLE

TITLE <text>

- ใช้กำหนด title ของ listing
- สูงสุด 60 characters

SUBTITLE

SUBTITLE <text>

- ใช้กำหนด subtitle
- สูงสุด 60 characters, ใช้เมื่อ title เขียนไม่พอ

Addressing Mode

- เป็นชนิดการเข้าถึงข้อมูลของ operand
- มี 7ชนิด และใน instruction เดียวกัน addressing mode ต่างกันได้

Addressing mode	Operand format	Segment register	example
Register addressing	register ยกเว้น IP	-	AX
Immediate addressing	constant value	-	2
Direct addressing	symbolic reference	DS	TABLE TABLE+2
Register indirect addressing	BX, DI, SI BP	DS SS	[BX] [BP]
Base relative addressing	[BX] + displacement [BP] + displacement	DS SS	[BX] + 4 TABLE[BX]
Direct indexed addressing	[DI] + displacement [SI] + displacement	DS	[DI] + 2 TABLE[DI]
Base indexed addressing	[BX][SI] + displacement [BX][DI] + displacement [BP][SI] + displacement [BP][DI] + displacement	DS DS SS SS	[BX][DI] + 2 TABLE[BX][DI]

By Natthaphach Anuwattananon

Assembly Mnemonic

MOV

MOV <dest>,<source>

- copy ข้อมูลจาก source ไปยัง dest
- dest และ source ต้องมีขนาดเท่ากัน
- ไม่สามารถ copy จาก symbolic ไปยัง symbolic ได้ ต้องใช้ register คั่น
- ไม่สามารถ copy จาก symbolic ไปยัง segment register ได้ ต้องใช้ register คั่น
- ไม่สามารถ copy จาก segment register ไปยัง segment register ได้ ต้องใช้ register คั่น
- ห้ามให้ CS เป็น dest

PUSH

PUSH <source>

- เก็บ source ขนาด 16 bit ไว้ที่ stack
- SS จะเก็บที่อยู่เริ่มต้นของ stack
- SP จะเก็บที่อยู่ของ top of stack เมื่อ push : SP -= 2

POP

POP <dest>

- copy ข้อมูลจาก top of stack ไปยัง dest
- เมื่อ POP : SP += 2

XCHG (exchange)

XCHG <dest>, <source>

- สลับค่าของ destination และ source
- ใช้กับ segment register ไม่ได้
- dest และ source ต้องมีขนาดเท่ากัน

ADD

ADD <dest>,<source>

- dest = dest + source
- flag ที่ถูก set 1 0
  - CF มีการทดจาก bit สูงสุด
  - PF 8 bit หลัง มี 1 เป็นจำนวนคู่
  - AF เป็น 1 หรือ 0 ก็ได้
  - ZF ผลลัพธ์เป็น 0
  - SF sign bit ของผลลัพธ์เป็น 1
  - OF ผลลัพธ์เกิด overflow
- overflow 8 bit
  - bit 6 7 ถ้าทดต้องทดทั้งคู่ ถ้าไม่ทดก็ต้องไม่ทดทั้งคู่ ถึงจะไม่เกิด overflow

SUB

SUB <dest>,<source>

- dest = dest - source
- flag ที่ถูก set 1 0
  - CF ไม่มีการทดจาก bit สูงสุด
  - PF 8 bit หลัง มี 1 เป็นจำนวนคู่
  - ZF ผลลัพธ์เป็น 0

By Natthaphach Anuwattananon

SF ผลลัพธ์เป็นลบ
OF เกิด overflow

MUL

MUL <source>

- ใช้กับ unsigned integer
- ถ้า source เป็น byte คูณกับ AL เก็บไว้ที่ AH AL
- ถ้า source เป็น word คูณกับ AX เก็บไว้ที่ DX AX
- flag ที่ถูก set 1 0
  - CF high-order half เป็น 0
  - OF high-order half เป็น 0

IMUL

IMUL <source>

- ใช้กับ signed integer
- ถ้า source เป็น byte คูณกับ AL เก็บไว้ที่ AH AL
- ถ้า source เป็น word คูณกับ AX เก็บไว้ที่ DX AX
- flag ที่ถูก set 1 0
  - CF high-order half เป็น sign extension ของ low-order half
  - OF high-order half เป็น sign extension ของ low-order half

DIV

DIV <source>

- ใช้กับ unsigned integer
- ถ้า source เป็น byte ตัวตั้งคือ AX เก็บผลที่ AL เก็บเศษที่ AH
- ถ้า source เป็น word ตัวตั้งคือ DX AX เก็บผลที่ AX เก็บเศษที่ DX
- ไม่มีการ set flag แต่จะเกิด interrupt เมื่อ
  - source = 0
  - ถ้า source เป็น byte ผลลัพธ์ตั้งแต่ 256 ขึ้นไป
  - ถ้า source เป็น word ผลลัพธ์ตั้งแต่ 65536 ขึ้นไป

IDIV

IDIV <source>

- ใช้กับ signed integer
- ถ้า source เป็น byte ตัวตั้งคือ AX เก็บผลที่ AL เก็บเศษที่ AH
- ถ้า source เป็น word ตัวตั้งคือ DX AX เก็บผลที่ AX เก็บเศษที่ DX
- ไม่มีการ set flag แต่จะเกิด interrupt เมื่อ
  - source = 0
  - ถ้า source เป็น byte ผลลัพธ์อยู่นอกช่วง -128 ถึง 127
  - ถ้า source เป็น word ผลลัพธ์อยู่นอกช่วง -32768 ถึง 32767

NEG

NEG <dest>

- ใช้หาค่าลบของ dest
- flag ที่ถูก set 1 0
  - CF dest เป็น nonzero positive
  - SF dest เป็น nonzero positive
  - PF ผลลัพธ์มี 1 เป็นจำนวนคู่

By Natthaphach Anuwattananon

ZF dest เป็น 0
OF dest เป็น 80H หรือ 8000H

CBW (convert byte to word)

CBW

- ใช้ขยาย AL ให้เป็น AX
- ทำให้ sign bit ของ AL กลายเป็นทุก bit ของ AH

CWD (convert word to doubleword)

CWD

- ใช้ขยาย AX ให้เป็น DX AX
- ทำให้ sign bit ของ AX กลายเป็นทุก bit ของ DX

DEC (decrement destination by one)

DEC <dest>

- dest = dest - 1

INC (increment destination by one)

INC <dest>

- dest = dest + 1

CMP (compare destination to source)

CMP <dest>,<source>

- flag ที่ถูก set

- unsigned	ZF	CF	
- dest > source	0	0	
- dest = source	1	0	
- dest < source	0	1	
- singed	OF	SF	ZF
- dest > source	0/1	0	0
- dest = source	0	0	1
- dest < source	0/1	1	0

Control transfer instructions

unconditional transfer

JMP

JMP <label>

conditional transfer

<J-mnemonic> <label>

J	Jump if	A	above	B	below	N	not
E	equal	C	carry	Z	zero	CX	CX
G	greater	L	less	O	overflow	P	parity
S	sign	unsigned ใช้ above, below			signed ใช้ greater, less		

LOOP

LOOP <label>

- ถ้าค่าใน CX != 0 จะทำการ jump ไปยัง label และ dec CX

PTR

<type> PTR <expression>

<type> ::= BYTE | WORD

- ใช้เพื่อระบุขนาดของ expression

LEA (Load Effective Address)

LEA <register>,<memory-location>

- ใช้เพื่อ load ค่า offset ของ memory location มาไว้ใน register

Interrupt intructions

INT

INT <interrupt-type>

<interrupt-type> ::= 0 ; divide error
1 ; single-step
2 ; nonmaskable interrupt
3 ; breakpoint
21H ; output

ขั้นตอนการทำงานของ INT

1. push flag register
2. clear TF, IF
3. push CS
4. หา interrupt-vector จาก interrupt-type
5. Load second word ของ interrupt-vector ลงใน CS
6. Push IP
7. Load first word ของ interrupt-vector ลงใน IP

IRET

- อยู่ใน interrupt routine ทำงานโดย

POP IP
POP CS
POP flag register

INT 21H

AH = 2	display character	DL	
AH = 9	display string	DX	(end with $)
AH = 4CH			

Procedure

CALL

CALL <procedure>

- เรียกใช้ procedure

RET

RET n

- กลับไปยังจุดที่เรียก และ pop ค่าใน stack ทิ้งไป n ค่า

By Natthaphach Anuwattananon

การส่งค่าไป-กลับ procedure
1. ส่งผ่าน register
2. ส่งผ่าน memory location
3. ส่งผ่าน stack

Logical Instructions

AND

AND <dest>,<source>

- dest = dest & source
- flag ที่ถูก set 1 0
  - OF เป็น 0 ตลอด
  - CF เป็น 0 ตลอด
  - AF undefined
  - SF sign bit เป็น 1
  - ZF มีค่าเป็น 0
  - PF มี 1 คู่ตัว

OR

OR <dest>,<source>

- dest = dest | source

XOR

XOR <dest>,<source>

- dest = dest ^ source

NOT

NOT <dest>

- dest = ~dest

TEST

TEST <dest>,<source>

- dest & source

	OF	DF	IF	TF	SF	ZF	AF	PF	CF
MOV									
PUSH									
POP									
XCHG									
ADD	1 | 0				1 | 0	1 | 0	1 | 0	1 | 0	1 | 0
SUB	1 | 0				1 | 0	1 | 0		1 | 0	1 | 0
MUL	1 | 0								1 | 0
IMUL	1 | 0								1 | 0
DIV									
IDIV									
NEG	1 | 0				1 | 0	1 | 0		1 | 0	1 | 0
CBW									
CWD									
DEC									
INC									
CMP	1 | 0				1 | 0	1 | 0	?	?	1 | 0
JMP									
LOOP									
PTR									
LEA									
AND	0				1 | 0	1 | 0	?	1 | 0	0
OR	0				1 | 0	1 | 0	?	1 | 0	0
XOR	0				1 | 0	1 | 0	?	1 | 0	0
NOT									

By Natthaphach Anuwattananon

Assembling program and debug

Load program

- เมื่อเริ่มโปรแกรม ต้อง load data segment และ extra segment เข้าไปยัง register

```
MOV   AX, DSEG        |    MOV   AX, ESEG
MOV   DS, AX          |    MOV   ES, AX
```

Debuging

T

```
T [n]
```

- ทำงาน n คำสั่ง (default n=1)

P

```
P [n]
```

- ทำงาน n คำสั่ง (ไม่เข้า procedure call, loop)

D

```
D <seg>:<start> [<end> | L<count>]
```

- เรียกดู segment ตั้งแต่ตำแหน่ง start
  - ถึงตำแหน่ง end
  - นับไปอีก count bytes

R

```
R [<register> | {<register> <value>}]
```

- เรียกดู register
  - ใส่ค่า value ให้กับ register

Q

```
Q
```

- ออกจาก debugging

By Natthaphach Anuwattananon