

CS231 ระบบคอมพิวเตอร์และภาษาแอสเซมบลี

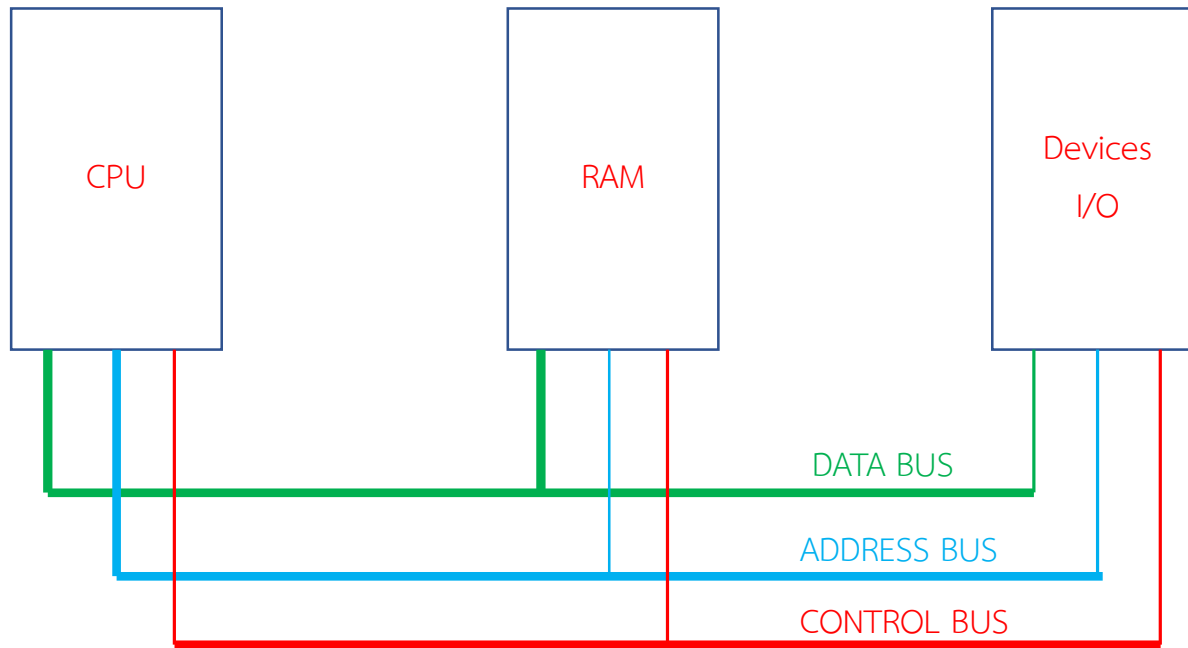
บทที่ 1: ภาษาแอสเซมบลี

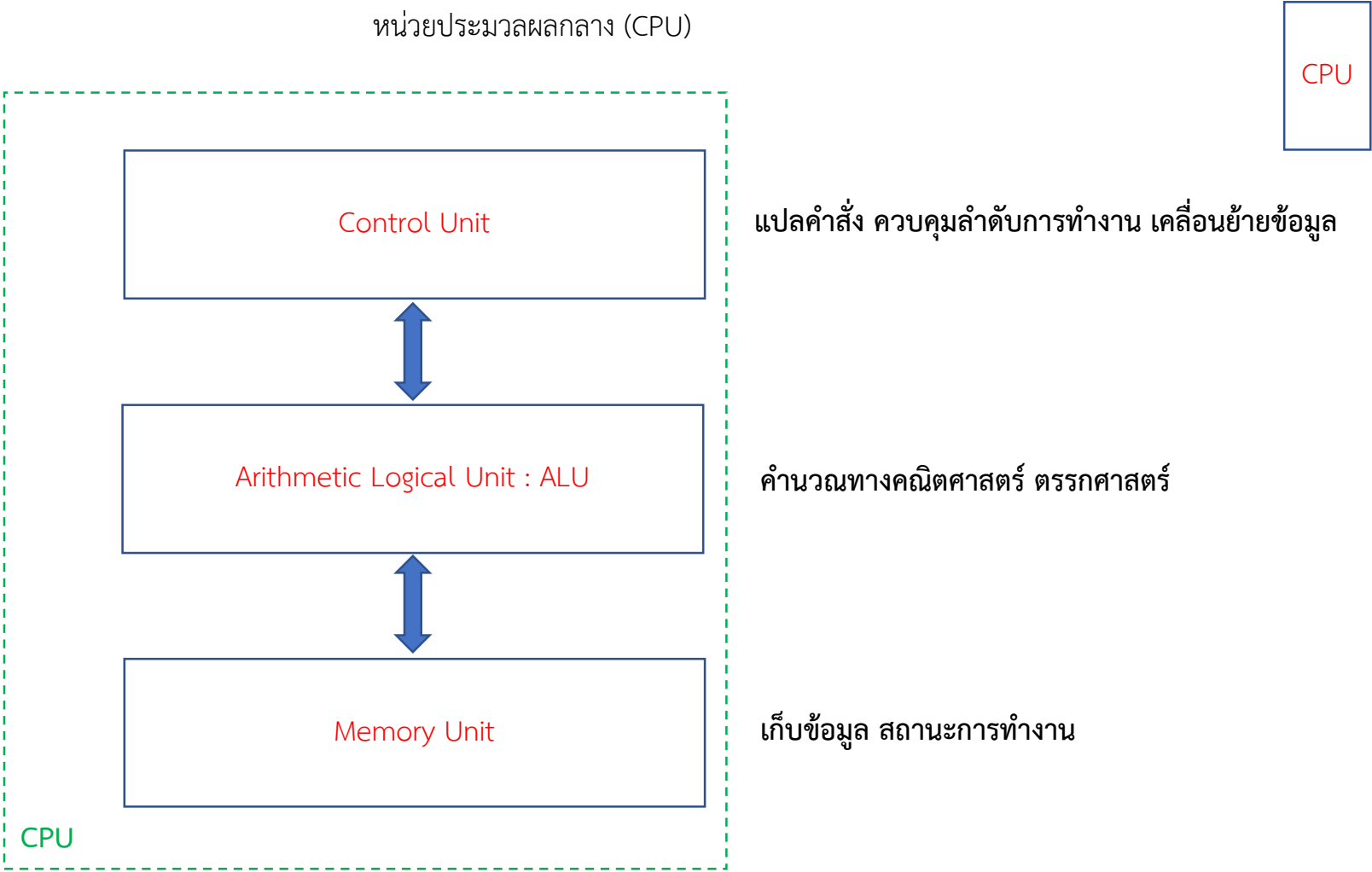


ผู้ช่วยศาสตราจารย์ ดร. ปวีณ เชื้อนแก้ว
สาขาวิชาวิทยาการคอมพิวเตอร์
คณะวิทยาศาสตร์ มหาวิทยาลัยแม่โจ้



ระบบคอมพิวเตอร์ (Computer System)





หน่วยประมวลผลกลาง (CPU): รีจิสเตอร์

หน่วยประมวลผลกลางบันทึกข้อมูลต่าง ๆ ไว้ในหน่วยความจำขนาดเล็ก เรียกว่า รีจิสเตอร์ (Register)

ขนาดของรีจิสเตอร์เป็นตัวกำหนดความสามารถในการประมวลผล ตัวอย่างเช่น

หน่วยประมวลผลขนาด 8 บิต จะมีรีจิสเตอร์ภายในขนาด 8 บิต จึงประมวลผลได้ที่ละ 8 บิต

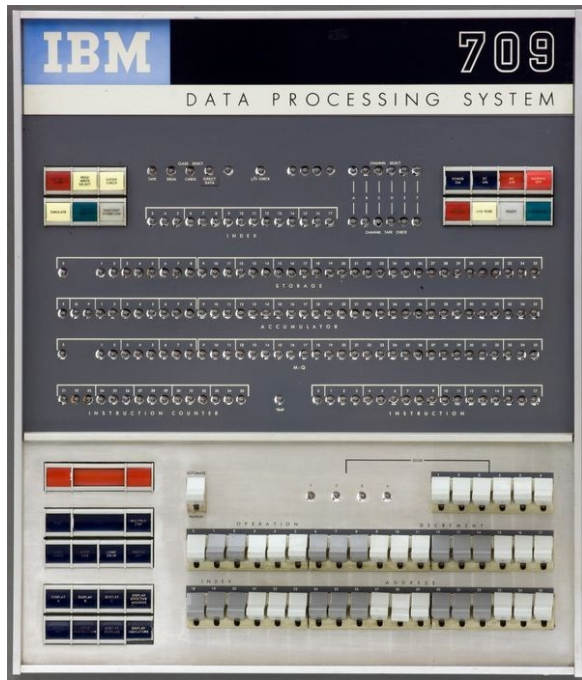
CPU

CPU แต่ละแบบจะมีจำนวนของรีจิสเตอร์แตกต่างกัน

รีจิสเตอร์ นิยมตั้งชื่อเรียกเป็น A, B, C, D

รีจิสเตอร์บางตัวอาจมีชื่อเรียกหลายชื่อ

รีจิสเตอร์บางตัวมีหน้าที่พิเศษในการทำงาน และมีบางตัวที่ถูกซ่อนไว้ หรือไม่สามารถโปรแกรมได้โดยตรง



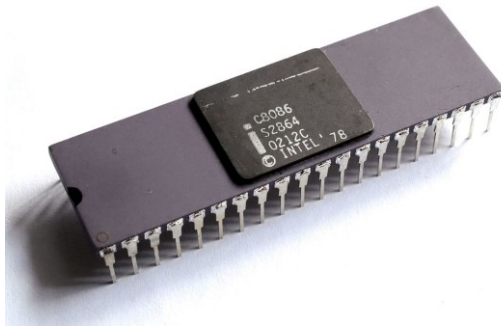
The IBM 709 was the last of IBM's large, scientific, vacuum-tube computers. Ease of programming and high-speed input/output were distinctive features. IBM announced a transistor-based successor in late 1958.

<https://www.computerhistory.org/revolution/early-computer-companies/5/111/1504?position=0>

หน่วยประมวลผลกลาง (CPU): รีจิสเตอร์

รีจิสเตอร์ของหน่วยประมวลผลกลาง 8086 มีทั้งหมด 14 ตัว

- สำหรับใช้งานทั่วไป 8 ตัว
- สำหรับระบุ เซกเมนต์ (ตำแหน่งหน่วยความจำภายนอก) 4 ตัว
- รีจิสเตอร์พิเศษ 2 ตัว



AL	AL
BH	BL
CH	CL
DH	DL
SP	
BP	
SI	
DI	
CS	
DS	
SS	
ES	
IP	
FLAGS	

หน่วยประมวลผลกลาง (CPU): รีจิสเตอร์

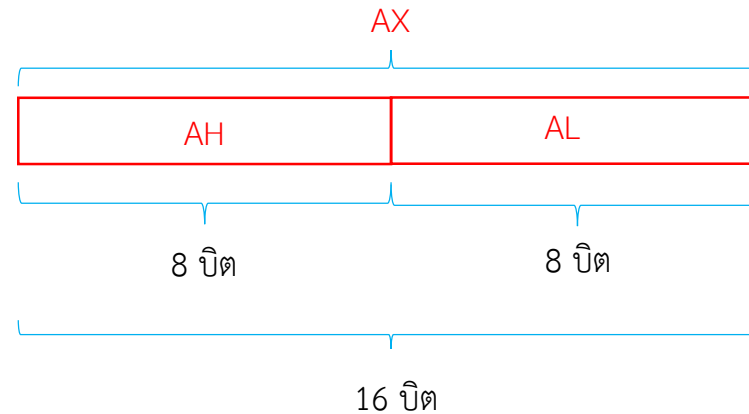
รีจิสเตอร์ของหน่วยประมวลผลกลาง 8086 สำหรับใช้งานทั่วไป 8 ตัว ประกอบไปด้วย

AX – Accumulator register ใช้สำหรับคำนวณ หรือใช้กับ ALU

มีขนาด 16 บิต (2 ไบต์)

8 บิตบน มีชื่อเรียกอีกชื่อว่า AH – Accumulator high

8 บิตล่าง มีชื่อเรียกอีกชื่อว่า AL – Accumulator low



หน่วยประมวลผลกลาง (CPU): รีจิสเตอร์

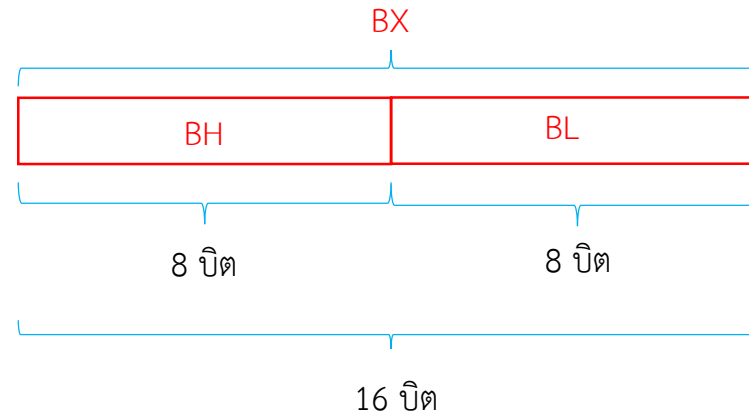
รีจิสเตอร์ของหน่วยประมวลผลกลาง 8086 สำหรับใช้งานทั่วไป 8 ตัว ประกอบไปด้วย

BX – Base address register ใช้สำหรับระบุตำแหน่งหน่วยความจำ

มีขนาด 16 บิต (2 ไบต์)

8 บิตบน มีชื่อเรียกอีกชื่อว่า BH

8 บิตล่าง มีชื่อเรียกอีกชื่อว่า BL



หน่วยประมวลผลกลาง (CPU): รีจิสเตอร์

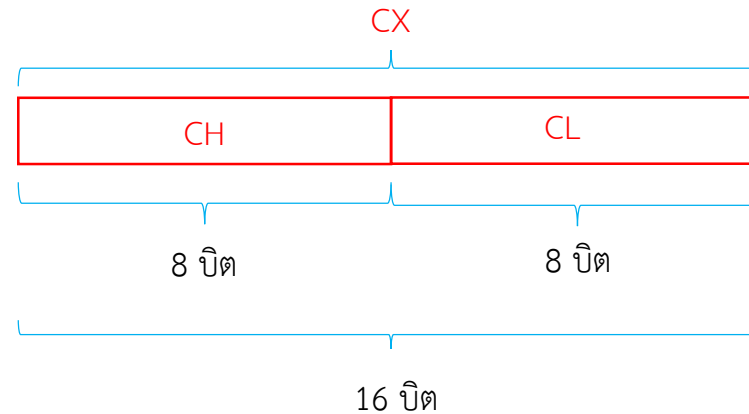
รีจิสเตอร์ของหน่วยประมวลผลกลาง 8086 สำหรับใช้งานทั่วไป 8 ตัว ประกอบไปด้วย

CX – Count register ใช้สำหรับนับ หรือใช้ในการวนซ้ำ

มีขนาด 16 บิต (2 ไบต์)

8 บิตบน มีชื่อเรียกอีกชื่อว่า CH

8 บิตล่าง มีชื่อเรียกอีกชื่อว่า CL



หน่วยประมวลผลกลาง (CPU): รีจิสเตอร์

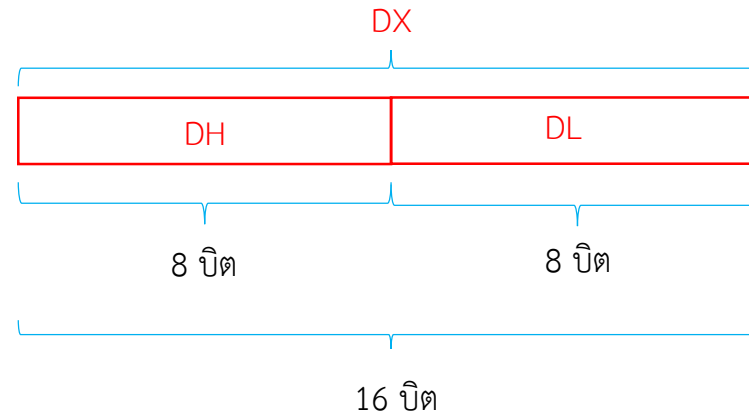
รีจิสเตอร์ของหน่วยประมวลผลกลาง 8086 สำหรับใช้งานทั่วไป 8 ตัว ประกอบไปด้วย

DX – Data register ใช้สำหรับเก็บข้อมูล

มีขนาด 16 บิต (2 ไบต์)

8 บิตบน มีชื่อเรียกอีกชื่อว่า DH

8 บิตล่าง มีชื่อเรียกอีกชื่อว่า DL

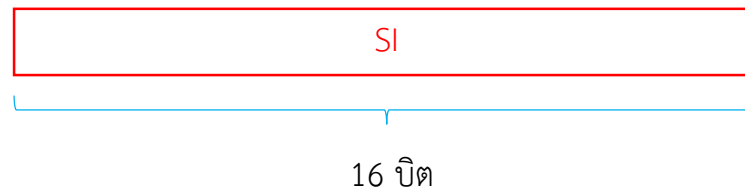


หน่วยประมวลผลกลาง (CPU): รีจิสเตอร์

รีจิสเตอร์ของหน่วยประมวลผลกลาง 8086 สำหรับใช้งานทั่วไป 8 ตัว ประกอบไปด้วย

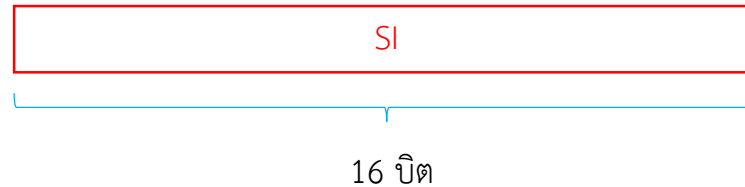
SI – Source index register ใช้สำหรับระบุตำแหน่ง Array

มีขนาด 16 บิต (2 ไบต์)



DI – Destination index register ใช้สำหรับระบุตำแหน่ง Array

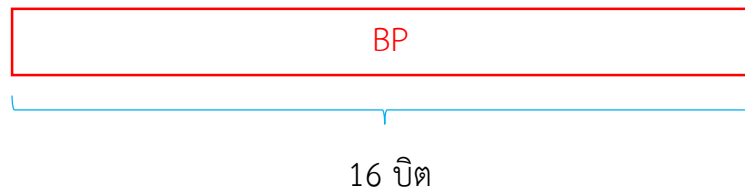
มีขนาด 16 บิต (2 ไบต์)



หน่วยประมวลผลกลาง (CPU): รีจิสเตอร์

รีจิสเตอร์ของหน่วยประมวลผลกลาง 8086 สำหรับใช้งานทั่วไป 8 ตัว ประกอบไปด้วย

BP – Base pointer register ใช้ชี้ตำแหน่งหน่วยความจำ
มีขนาด 16 บิต (2 ไบต์)



SP – Stack pointer register ใช้สำหรับโครงสร้างข้อมูลแบบแถวซ้อน
มีขนาด 16 บิต (2 ไบต์)

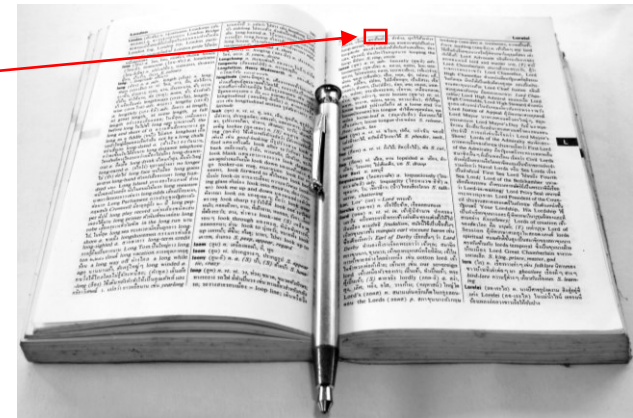


หน่วยประมวลผลกลาง (CPU): รีจิสเตอร์

รีจิสเตอร์ของหน่วยประมวลผลกลาง 8086 สำหรับใช้งานระบุ Segment 4 ตัว ประกอบไปด้วย

การระบุตำแหน่งด้วยเซกเมนต์ ช่วยให้เขียนโปรแกรมได้ง่ายขึ้น เพราะหน่วยความจำถูกแบ่งเป็นส่วน ๆ

หน้าที่ 100 ตัวอักษรที่ 10
เซกเมนต์ ออฟเซต



- CS – Code Segment ใช้ระบุ เซกเมนต์ ตำแหน่งโปรแกรมในหน่วยความจำ มีขนาด 16 บิต
- DS – Data Segment ใช้ระบุ เซกเมนต์ ตำแหน่งข้อมูลหรือตัวแปรในหน่วยความจำ มีขนาด 16 บิต
- ES – Extra Segment ใช้ระบุ เซกเมนต์ ตำแหน่งข้อมูลเพิ่มเติมในหน่วยความจำ มีขนาด 16 บิต
- SS – Stack Segment ใช้ระบุ เซกเมนต์ ตำแหน่งโครงสร้างข้อมูลแถวซ้อนในหน่วยความจำ มีขนาด 16 บิต

หน่วยประมวลผลกลาง (CPU): รีจิสเตอร์

รีจิสเตอร์ของหน่วยประมวลผลกลาง 8086 สำหรับใช้พิเศษ 2 ตัว ประกอบไปด้วย

IP – Instruction Pointer ใช้ระบุตำแหน่งของโปรแกรมที่กำลังรันอยู่ ใช้ควบคู่กับ CS มีขนาด 16 บิต
ค่าจะเปลี่ยนแปลงไปขณะที่โปรแกรมทำงาน

Flags ใช้เก็บสถานะภายในของหน่วยประมวลผล ตัวอย่างเช่น การเกิด overflow หรือผลการคำนวณติดลบ
ไม่สามารถเข้าถึงโดยตรงจากโปรแกรมได้

สิ่งที่ไมโครโพรเซสเซอร์เข้าใจ เรียกว่า Operation Code หรือ opcode
บางคำสั่งมีแค่ opcode ตัวอย่างเช่นคำสั่งหยุดการทำงาน
บางคำสั่งมีพารามิเตอร์อื่นประกอบ ตัวอย่างเช่นคำสั่ง ย้ายข้อมูล พารามิเตอร์ที่เกี่ยวข้องนี้เรียกว่า operand

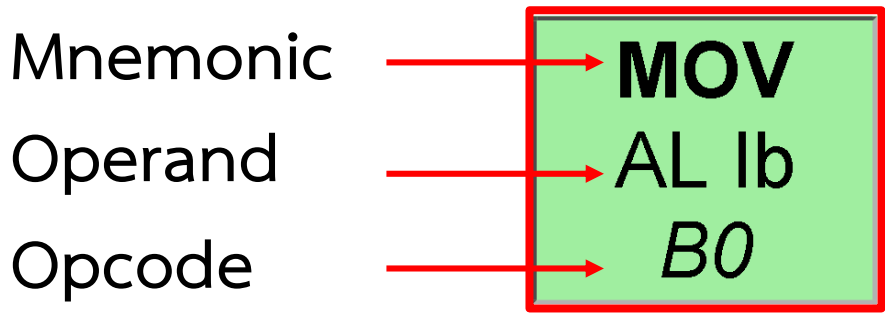
opcode operand

ตาราง opcode ของไมโครโพรเซสเซอร์ X86

ADD Eb Gb 00	ADD Ev Iv 01	ADD Gb Eb 02	ADD Gv Ev 03	ADD AL Ib 04	ADD eAX Iv 05	PUSH ES 06	POP ES 07	OR Eb Gb 08	OR Ev Iv 09	OR Gb Eb 0A	OR Gv Ev 0B	OR AL Ib 0C	OR eAX Iv 0D	PUSH CS 0E	TWOBYTE 0F
ADC Eb Gb 10	ADC Ev Iv 11	ADC Gb Eb 12	ADC Gv Ev 13	ADC AL Ib 14	ADC eAX Iv 15	PUSH SS 16	POP SS 17	SBB Eb Gb 18	SBB Ev Iv 19	SBB Gb Eb 1A	SBB Gv Ev 1B	SBB AL Ib 1C	SBB eAX Iv 1D	PUSH DS 1E	POP DS 1F
AND Eb Gb 20	AND Ev Iv 21	AND Gb Eb 22	AND Gv Ev 23	AND AL Ib 24	AND eAX Iv 25	ES: 26	DAA 27	SUB Eb Gb 28	SUB Ev Iv 29	SUB Gb Eb 2A	SUB Gv Ev 2B	SUB AL Ib 2C	SUB eAX Iv 2D	CS: 2E	DAS 2F
XOR Eb Gb 30	XOR Ev Iv 31	XOR Gb Eb 32	XOR Gv Ev 33	XOR AL Ib 34	XOR eAX Iv 35	SS: 36	AAA 37	CMP Eb Gb 38	CMP Ev Iv 39	CMP Gb Eb 3A	CMP Gv Ev 3B	CMP AL Ib 3C	CMP eAX Iv 3D	DS: 3E	AAS 3F
INC eAX 40	INC eCX 41	INC eDX 42	INC eBX 43	INC eSP 44	INC eBP 45	INC eSI 46	INC eDI 47	DEC eAX 48	DEC eCX 49	DEC eDX 4A	DEC eBX 4B	DEC eSP 4C	DEC eBP 4D	DEC eSI 4E	DEC eDI 4F
PUSH eAX 50	PUSH eCX 51	PUSH eDX 52	PUSH eBX 53	PUSH eSP 54	PUSH eBP 55	PUSH eSI 56	PUSH eDI 57	POP eAX 58	POP eCX 59	POP eDX 5A	POP eBX 5B	POP eSP 5C	POP eBP 5D	POP eSI 5E	POP eDI 5F
PUSHA 60	POPA 61	BOUND Gv Ma 62	ARPL Ew Gv 63	FS: 64	GS: 65	OPSIZE: 66	ADSIZE: 67	PUSH Iv 68	IMUL Gv Ev Iv 69	PUSH Ib 6A	IMUL Gv Ev Ib 6B	INSB Yb DX 6C	INSD Yz DX 6D	OUTSB DX Xb 6E	OUTSW DX Xv 6F
JO Jb 70	JNO Jb 71	JB Jb 72	JNB Jb 73	JZ Jb 74	JNZ Jb 75	JBE Jb 76	JA Jb 77	JJS Jb 78	JNS Jb 79	JP Jb 7A	JNP Jb 7B	JL Jb 7C	JNL Jb 7D	JLE Jb 7E	JNLE Jb 7F

ADD Eb Ib 80	ADD Ev Iv 81	SUB Eb Ib 82	SUB Ev Iv 83	TEST Eb Gb 84	TEST Ev Gv 85	XCHG Eb Gb 86	XCHG Ev Gv 87	MOV Eb Gb 88	MOV Ev Gv 89	MOV Gb Eb 8A	MOV Gv Ev 8B	MOV Ew Sw 8C	MOV Gv M 8D	LEA Gv Sw 8E	POP Sw Ev 8F
NOP 90	XCHG eAX eCX 91	XCHG eAX eDX 92	XCHG eAX eBX 93	XCHG eAX eSP 94	XCHG eAX eBP 95	XCHG eAX eSI 96	XCHG eAX eDI 97	CBW 98	CWD 99	CALL Ap 9A	WAIT 9B	PUSHF Fv 9C	POPF Fv 9D	SAHF 9E	LAHF 9F
MOV AL Ob A0	MOV eAX Ov A1	MOV Ob AL A2	MOV Ov eAX A3	MOVS Xb Yb A4	MOVS Xv Yv A5	CMP Xb Yb A6	CMP Xv Yv A7	TEST AL Ib A8	TEST eAX Iv A9	STOSB Yb AL AA	STOSW Yv eAX AB	LODSB AL Xb AC	LODSW eAX Xv AD	SCASB AL Yb AE	SCASW eAX Yv AF
MOV AL Ib B0	MOV CL Ib B1	MOV DL Ib B2	MOV BL Ib B3	MOV AH Ib B4	MOV CH Ib B5	MOV BH Ib B6	MOV DH Ib B7	MOV eAX Iv B8	MOV eCX Iv B9	MOV eDX Iv BA	MOV eBX Iv BB	MOV eSP Iv BC	MOV eBP Iv BD	MOV eSI Iv BE	MOV eDI Iv BF
#2 Eb Ib C0	#2 Ev Ib C1	RET Iw C2	RET C3	LES Gv Mp C4	LDS Gv Mp C5	MOV Eb Ib C6	MOV Ev Ib C7	ENTER Iw Ib C8	LEAVE C9	RETF Iw CA	RETF CB	INT3 CC	INT CD	INTO CE	IRET CF
#2 Eb 1 D0	#2 Ev 1 D1	#2 Eb CL D2	#2 Ev CL D3	AAM Ib D4	AAD Ib D5	SALC D6	XLAT D7	ESC 0 D8	ESC 1 D9	ESC 2 DA	ESC 3 DB	ESC 4 DC	ESC 5 DD	ESC 6 DE	ESC 7 DF
LOOPNZ Jb E0	LOOPZ Jb E1	LOOP Jb E2	JCJZ Jb E3	IN AL Ib E4	IN eAX Ib E5	OUT Ib AL E6	OUT Ib eAX E7	CALL Jz E8	JMP Jz E9	JMP Ap EA	JMP Eb EB	IN DX AL EC	IN eAX DX ED	OUT DX AL EE	OUT eAX DX EF
LOCK: F0	INT1 F1	REPNE: F2	REP: F3	HLT F4	CMC F5	#3 Eb F6	#3 Ev F7	CLC F8	STC F9	CLI FA	STI FB	CLD FC	STD FD	#4 INC/DEC FE	#5 INC/DEC FF

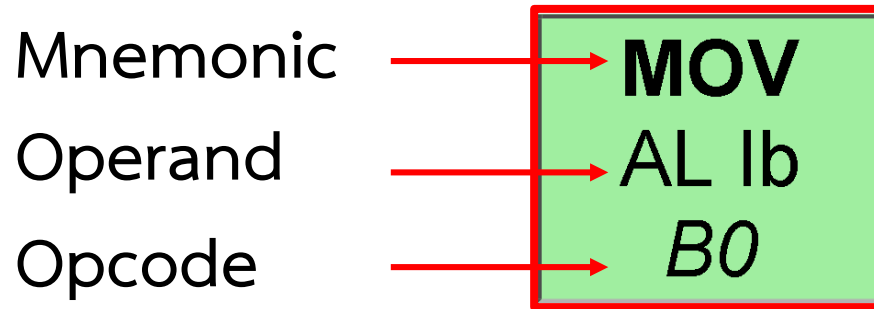
<http://sparksandflames.com/files/x86InstructionChart.html>



การโปรแกรมด้วย opcode คือการสั่งงานด้วยภาษาเครื่อง
ยาก ไม่เป็นที่นิยม ซึ่งเราจะมาศึกษาในหลัง midterm*

การโปรแกรมด้วย Mnemonic นั้นเป็นที่นิยมมากกว่า เพราะ
ไม่ต้องจำ opcode แต่เราต้องมีตาราง opcode เพื่อตรวจสอบว่า
CPU สามารถทำงานตามที่เราต้องการได้หรือไม่

* เนื้อหา Instruction Encoding อาจเปลี่ยนแปลงได้ขึ้นอยู่กับคะแนนสอบ midterm



ภาษาที่โปรแกรมด้วย Mnemonic คือ

ภาษา Assembly

โปรแกรมที่ใช้แปลภาษา Assembly เป็นภาษาเครื่อง คือ

โปรแกรม Assembler

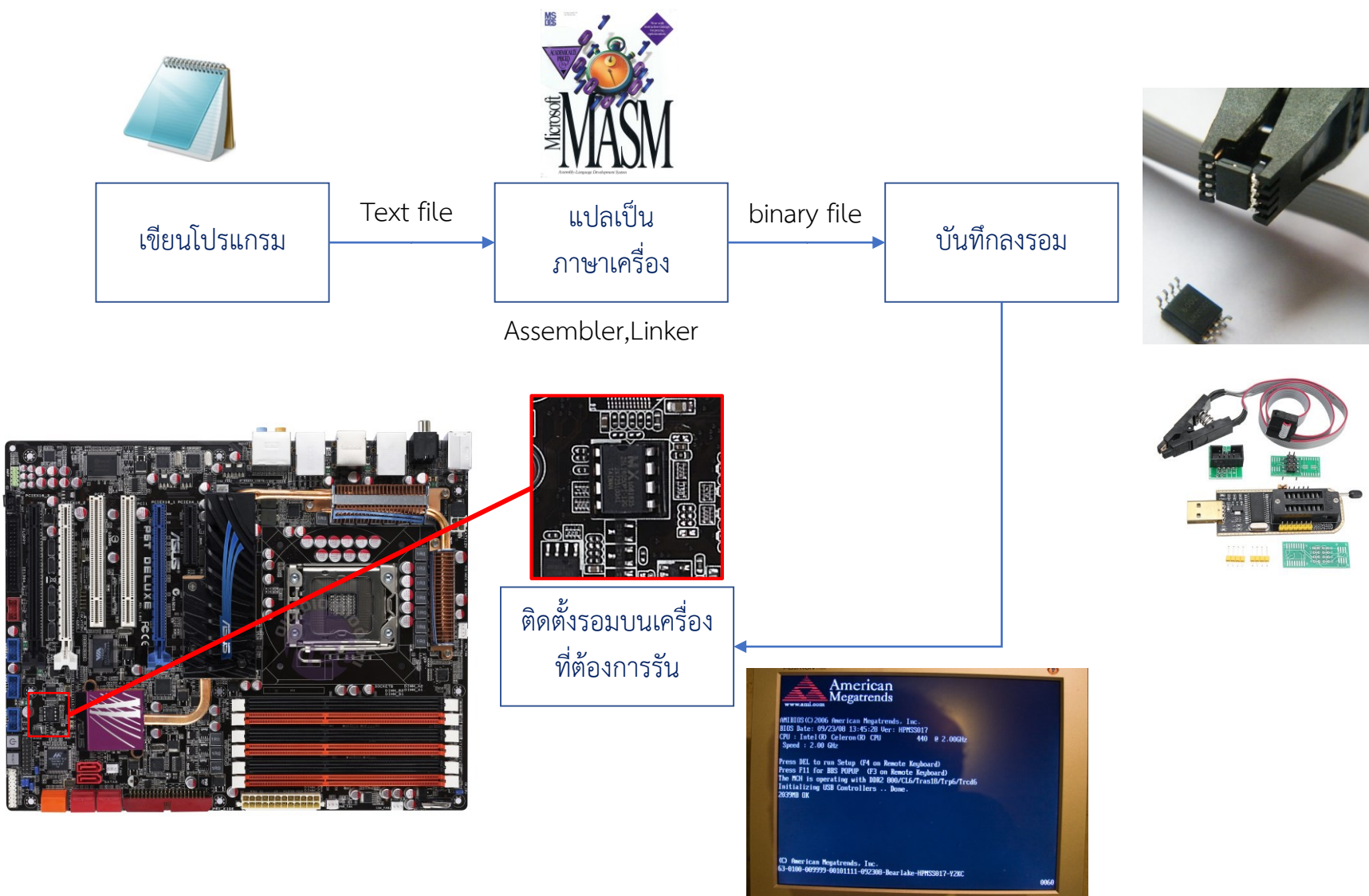
CS231: บทที่ 1 : ภาษาแอสเซมบลี

ADD Eb Gb 00	ADD Ev Gv 01	ADD Gb Eb 02	ADD Gv Ev 03	ADD AL Ib 04	ADD eAX Iv 05	PUSH ES 06	POP ES 07	OR Eb Gb 08	OR Ev Gv 09	OR Gb Eb 0A	OR Gv Ev 0B	OR AL Ib 0C	OR eAX Iv 0D	PUSH CS 0E	TWOBYTE 0F	ADD Eb Ib 80	ADD Ev Iv 81	SUB Eb Ib 82	SUB Ov Ib 83	TEST Eb Gb 84	TEST Ev Gv 85	XCHG Eb Gb 86	XCHG Ev Gv 87	MOV Eb Gb 88	MOV Ev Gv 89	MOV Gb AL 8A	MOV Gv Eb 8B	MOV Ew Sw 8C	LEA Gv M 8D	MOV Sw Ew 8E	POP Ev 8F
ADC Eb Gb 10	ADC Ev Gv 11	ADC Gb Eb 12	ADC Gv Ev 13	ADC AL Ib 14	ADC eAX Iv 15	PUSH SS 16	POP SS 17	SBB Eb Gb 18	SBB Ev Gv 19	SBB Gb Eb 1A	SBB Gv Ev 1B	SBB AL Ib 1C	SBB eAX Iv 1D	PUSH DS 1E	POP DS 1F	NOP 90	XCHG eAX eCX 91	XCHG eAX eDX 92	XCHG eAX eBX 93	XCHG eAX eSP 94	XCHG eAX eBP 95	XCHG eAX eSI 96	XCHG eAX eDI 97	CBW 98	CWD 99	CALL Ap 9A	WAIT 9B	PUSHF Fv 9C	POPF Fv 9D	SAHF 9E	LAHF 9F
AND Eb Gb 20	AND Ev Gv 21	AND Gb Eb 22	AND Gv Ev 23	AND AL Ib 24	AND eAX Iv 25	ES: 26	DAA 27	SUB Eb Gb 28	SUB Ev Gv 29	SUB Gb Eb 2A	SUB Gv Ev 2B	SUB AL Ib 2C	SUB eAX Iv 2D	CS: 2E	DAS 2F	MOV AL Ob A0	MOV eAX Ov A1	MOV Ob AL A2	MOV Ov eAX A3	MOVSB A4	MOVSW A5	CMPBSB A6	CMPBSW A7	TEST AL Ib A8	TEST eAX Iv A9	STOSB Yb AL AA	STOSW Yv eAX AB	LODSB AL Xb AC	LODSW eAX Xv AD	SCASB AL Yb AE	SCASW eAX Yv AF
XOR Eb Gb 30	XOR Ev Gv 31	XOR Gb Eb 32	XOR Gv Ev 33	XOR AL Ib 34	XOR eAX Iv 35	SS: 36	AAA 37	CMP Eb Gb 38	CMP Ev Gv 39	CMP Gb Eb 3A	CMP Gv Ev 3B	CMP AL Ib 3C	CMP eAX Iv 3D	DS: 3E	AAS 3F	MOV AL Ib B0	MOV CL Ib B1	MOV DL Ib B2	MOV BL Ib B3	MOV AH Ib B4	MOV CH Ib B5	MOV DH Ib B6	MOV BH Ib B7	MOV eAX Iv B8	MOV eCX Iv B9	MOV eDX Iv BA	MOV eBX Iv BB	MOV eSP Iv BC	MOV eBP Iv BD	MOV eSI Iv BE	MOV eDI Iv BF
INC eAX 40	INC eCX 41	INC eDX 42	INC eBX 43	INC eSP 44	INC eBP 45	INC eSI 46	INC eDI 47	DEC eAX 48	DEC eCX 49	DEC eDX 4A	DEC eBX 4B	DEC eSP 4C	DEC eBP 4D	DEC eSI 4E	DEC eDI 4F	#2 Eb Ib C0	#2 Ev Ib C1	RETN lw C2	RETN C3	LES Gv Mp C4	LDS Gv Mp C5	MOV Eb Ib C6	MOV Ev Iv C7	ENTER lw Ib C8	LEAVE C9	RETF lw CA	RETF CB	INT3 CC	INT lb CD	INTO CE	IRET CF
PUSH eAX 50	PUSH eCX 51	PUSH eDX 52	PUSH eBX 53	PUSH eSP 54	PUSH eBP 55	PUSH eSI 56	PUSH eDI 57	POP eAX 58	POP eCX 59	POP eDX 5A	POP eBX 5B	POP eSP 5C	POP eBP 5D	POP eSI 5E	POP eDI 5F	#2 Eb 1 D0	#2 Ev 1 D1	#2 Eb CL D2	#2 Ev CL D3	AAM lb D4	AAD lb D5	SALC D6	XLAT D7	ESC 0 D8	ESC 1 D9	ESC 2 DA	ESC 3 DB	ESC 4 DC	ESC 5 DD	ESC 6 DE	ESC 7 DF
PUSHA 60	POPA 61	BOUND Gv Ma 62	ARPL Ew Gv 63	FS: 64	GS: 65	OPSIZE: 66	ADSIZE: 67	PUSH iv 68	IMUL Gv Ev Iv 69	PUSH lb 6A	IMUL Gv Ev Ib 6B	INSB Yb DX 6C	INSD Yz DX 6D	OUTSB DX Xb 6E	OUTSW DX Xv 6F	LOOPNZ Jb E0	LOOPZ Jb E1	LOOP E2	JCJZ E3	IN E4	IN eAX Ib E5	OUT lb AL E6	OUT lb eAX E7	CALL Jz E8	JMP Jz EA	JMP Ap EB	JMP Jb EC	IN AL DX ED	IN eAX DX EE	OUT DX AL EF	OUT DX eAX EF
JO Jb 70	JNO Jb 71	JB Jb 72	JNB Jb 73	JZ Jb 74	JNZ Jb 75	JBE Jb 76	JA Jb 77	JNS Jb 78	JJS Jb 79	JP Jb 7A	JNP Jb 7B	JL Jb 7C	JNL Jb 7D	JLE Jb 7E	JNLE Jb 7F	LOCK: F0	INT1 F1	REPNE: F2	REP: F3	HLT F4	CMC F5	#3 Eb F6	#3 Ev F7	CLC F8	STC F9	CLI FA	STI FB	CLD FC	STD FD	#4 INC/DEC FE	#5 INC/DEC FF

AA	CMPSB	JAE	JNBE	JPO	MOVSB	RCR	SCASB
AAD	CMPSW	JB	JNC	JS	MOVSW	REP	SCASW
AAM	DAA	JBE	JNE	JZ	MUL	REPE	SHL
AAS	DAS	JC	JNG	LAHF	NEG	REPNE	SHR
ADC	DEC	JCXZ	JNGE	LDS	NOP	REPNZ	STC
ADD	DIV	JE	JNL	LEA	NOT	REPZ	STD
AND	HLT	JG	JNLE	LES	OR	RET	STI
CALL	IDIV	JGE	JNQ	LODSB	OUT	RETF	STOSB
CBW	IMUL	JL	JNP	LODSW	POP	ROL	STOSW
CLC	IN	JLE	JNS	LOOP	POPA	ROR	SUB
CLD	INC	JMP	JNZ	LOOPE	POPF	SAHF	TEST
CLI	INT	JNA	JO	LOOPNE	PUSH	SAL	XCHG
CMC	INTO	JNAE	JP	LOOPNZ	PUSHA	SAR	XLATB
CMP	IRET	JNB	JPE	LOOPZ	PUSHF	SBB	XOR
JA					RCL		

จำคำสั่งน้อยลง เมื่อโปรแกรมด้วยภาษา Assembly

Work flow ของการโปรแกรมด้วยภาษา Assembly (แบบฝังตัว)



Work flow ของการโปรแกรมด้วยภาษา Assembly (แบบโปรแกรมประยุกต์)



เราจะเริ่มศึกษาในรูปแบบของโปรแกรมฝังตัว

- ไม่มี Bios
- ไม่มีระบบปฏิบัติการ
- เปิดเครื่องมาเพื่อรันโปรแกรมเดียว (เหมือน Arduino)



จากนั้นค่อยไปสร้างโปรแกรมประยุกต์บนแพลตฟอร์ม พีซี+ดอส และ พีซี+วินโดวส์



Work flow ของการโปรแกรมด้วยภาษา Assembly (แบบฝังตัว ด้วยโปรแกรมจำลอง)



```
edit: C:\emu8086\examples\1_sample.asm
file edit bookmarks assembler emulator math ascii codes help
new open examples save compile emulate calculate
01 name "hi-world"
02
03 ; this example prints out
04 ; by writing directly to vi
05 ; in vga memory: first byte
06 ; if you change the second
07 ; the character even after
08 ; character attribute is 8
09 ; high 4 bits set backgrou
10
11 ; hex      bin      color
12 ; 0        0000     black
13 ; 1        0001     blue
14 ; 2        0010     green
15
```

emulator: hi-world.com_

file math debug view external virtual devices virtual drive help

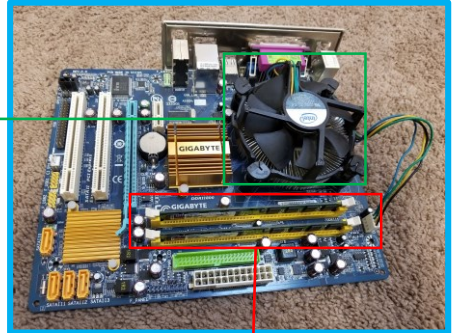
Load reload step back single step run step delay ms: 0

registers	H	L
AX	00	00
BX	00	00
CX	00	61
DX	00	00
CS	0700	
IP	0100	
SS	0700	
SP	FFFE	
BP	0000	
SI	0000	
DI	0000	
DS	0700	
ES	0700	

0700:0100

Address	Hex	Bin	Op	Comment
07100:	B8 184		MOV	AX, 00003h
07101:	03 003		INT	010h
07102:	00 000		MOV	AX, 01003h
07103:	CD 205		MOV	BX, 00000h
07104:	10 016		INT	010h
07105:	B8 184		MOV	AX, 0B800h
07106:	03 003		MOV	DS, AX
07107:	10 016		MOV	b:[00002h], 048h
07108:	B8 187		MOV	b:[00004h], 065h
07109:	00 000		MOV	b:[00006h], 06Ch
0710A:	00 000		MOV	b:[00008h], 06Ch
0710B:	CD 205		MOV	b:[0000Ah], 06Fh
0710C:	10 016			
0710D:	B8 184			
0710E:	00 000			
0710F:	B8 184			
07110:	8E 142			
07111:	08 216			
07112:	C6 198			
07113:	06 006			
07114:	02 002			
07115:	00 000			

screen source reset aux vars debug stack flags



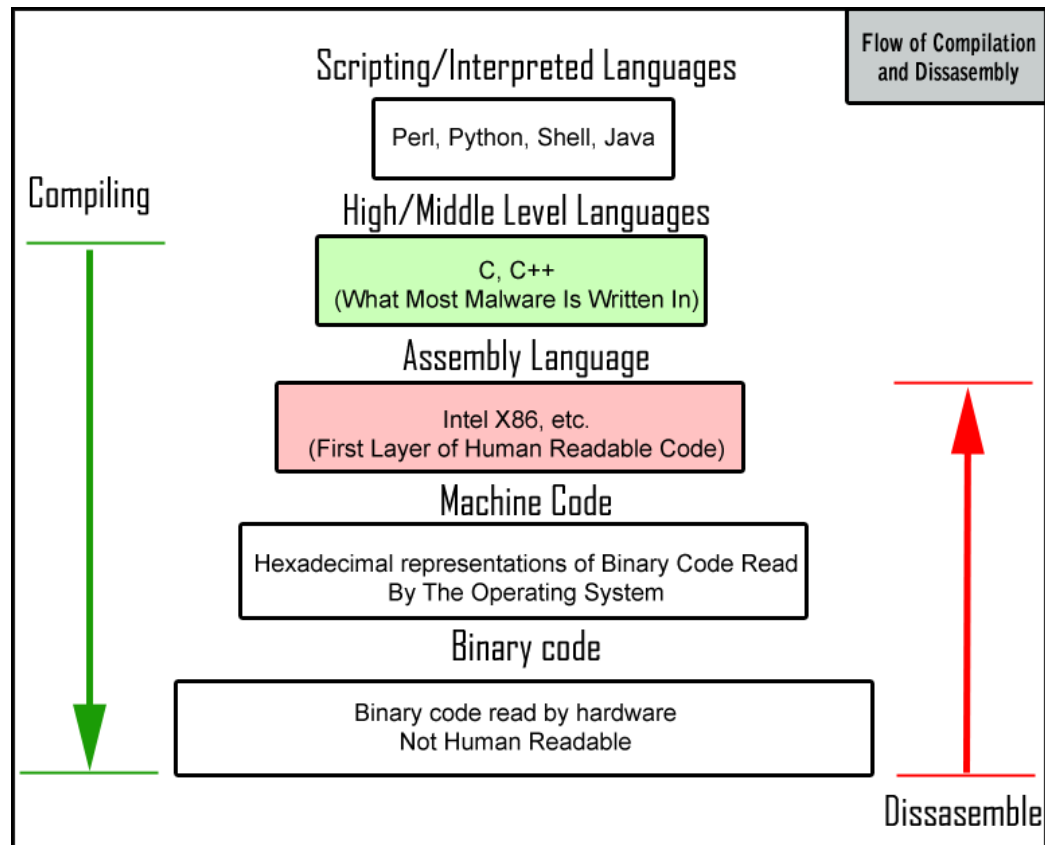
Random Access Memory

0700:0100 update table list

Address	Hex	Bin	Op	Comment
0700:0100	B8 03 00		MOV	AX, 00003h
0700:0101	03 003		INT	010h
0700:0102	00 000		MOV	AX, 01003h
0700:0103	CD 205		MOV	BX, 00000h
0700:0104	10 016		INT	010h
0700:0105	B8 184		MOV	AX, 0B800h
0700:0106	03 003		MOV	DS, AX
0700:0107	10 016		MOV	b:[00002h], 048h
0700:0108	B8 187		MOV	b:[00004h], 065h
0700:0109	00 000		MOV	b:[00006h], 06Ch
0700:010A	00 000		MOV	b:[00008h], 06Ch
0700:010B	CD 205		MOV	b:[0000Ah], 06Fh
0700:010C	10 016			
0700:010D	B8 184			
0700:010E	00 000			
0700:010F	B8 184			
0700:0110	8E 142			
0700:0111	08 216			
0700:0112	C6 198			
0700:0113	06 006			
0700:0114	02 002			
0700:0115	00 000			

ภาษาแอสเซมบลี (Assembly Language)

เป็นภาษาคอมพิวเตอร์ เรียกว่า ASM เป็นภาษาระดับต่ำ ที่มีความสัมพันธ์ยืดยึดกับ คำสั่ง สถาปัตยกรรม และชนิดของหน่วยประมวลผล



<https://blog.malwarebytes.com/security-world/2012/09/so-you-want-to-be-a-malware-analyst/#>

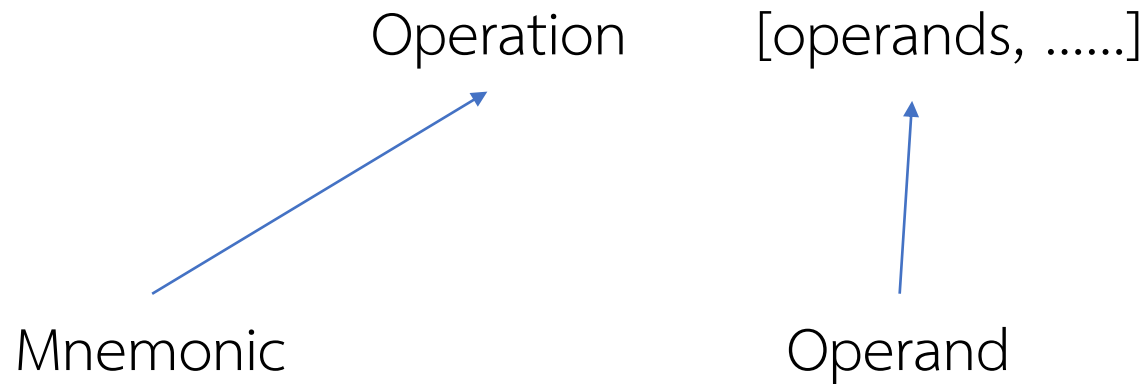
ภาษาแอสเซมบลี (Assembly Language)

เป็นภาษาคอมพิวเตอร์ เรียกย่อ ๆ ว่า ASM เป็นภาษาระดับต่ำ ที่มีความสัมพันธ์ยืดยึดติดกับ คำสั่งสถาปัตยกรรม และชนิดของหน่วยประมวลผล

- มี Abstraction น้อยมาก
- ภาษาระดับต่ำที่สุดที่มนุษย์สามารถอ่านได้
- สามารถควบคุม hardware ได้ในระดับต่ำที่สุด
- โปรแกรมยาว
- ไม่สามารถนำไปรันต่างแพลตฟอร์มได้
- ทำงานเร็วมาก

ภาษาแอสเซมบลี (Assembly Language)

เริ่มทำงานจากบรรทัดแรกจนถึงบรรทัดสุดท้าย รูปแบบคำสั่งจะประกอบด้วย 2 ส่วนคือ



เป็นภาษาที่ไม่มี case sensitivity

helloworld == HELLoworld

ภาษาแอสเซมบลี (Assembly Language)

การโปรแกรม นิยมใช้ตัวเลขฐาน 16 เพราะง่ายในการแปลง แต่หากไม่มีการเติม suffix โปรแกรม assembler จะมองตัวเลข decimal (0 – 9) ทุกตัวเป็นเลขฐาน 10

Suffix	ความหมาย	ตัวอย่าง
ไม่มี	เลขฐาน 10	10
b	เลขฐาน 2	10b
o	เลขฐาน 8	10o
h	เลขฐาน 16	10H

การป้อนตัวเลขที่ฐาน 16 นั้นอยู่นอกช่วงของ decimal จะต้องใช้ prefix 0x นำหน้าจำนวน และไม่ต้องเติม suffix ตัวอย่างเช่น

0x FA จะมีค่าเท่ากับ 250 ในเลขฐาน 10

ถ้าป้อนเพียง FAH จะคอมไพล์ไม่ผ่าน

ภาษาแอสเซมบลี (Assembly Language)

การโปรแกรม นิยมใช้ตัวเลขฐาน 16 เพราะง่ายในการแปลผล แต่หากไม่มีการเติม suffix โปรแกรม assembler จะมองตัวเลข decimal (0 – 9) ทุกตัวเป็นเลขฐาน 10

Suffix	ความหมาย	ตัวอย่าง
ไม่มี	เลขฐาน 10	10
b	เลขฐาน 2	10b
o	เลขฐาน 8	10o
h	เลขฐาน 16	10H

การป้อนตัวเลขที่ฐาน 16 นั้นอยู่นอกช่วงของ decimal จะต้องใช้ prefix 0x นำหน้าจำนวน และไม่ต้องเติม suffix ตัวอย่างเช่น

0x FA จะมีค่าเท่ากับ 250 ในเลขฐาน 10

ถ้าป้อนเพียง FAH จะคอมไพล์ไม่ผ่าน

ภาษาแอสเซมบลี (Assembly Language)

Operation [operands,]

Operation คือคำสั่งการทำงาน จะแตกต่างกันตามหน่วยประมวลผล

Operand จะมีเพียง 4 ประเภทคือ

operand	ความหมาย	ตัวอย่าง
REG	รีจิสเตอร์ AX, BX, CX, DH, SP, BP, SI, DI	MOV AX, BX
SREG	รีจิสเตอร์ segment CS, DS, SS, ES	MOV DS, AX
Immediate	ค่าใช้ทันที	MOV AX, 1234
Memory	ตำแหน่งที่อยู่ในหน่วยความจำ ระบุด้วยเครื่องหมาย []	MOV AX, [1234] MOV AX, [CP]

ภาษาแอสเซมบลี (Assembly Language)

ไมโครโพรเซสเซอร์แต่ละรุ่น จะมี opcode แตกต่างกัน
การโปรแกรมจำเป็นต้องใช้คู่มือจากผู้ผลิต

จากนี้ไปจะกล่าวถึง opcode ของไมโครโพรเซสเซอร์ intel 8086

