

**MACHINE LANGUAGE
CODING AND THE DEBUG
SOFTWARE DEVELOPMENT
PROGRAM OF THE PC**

A slide content graphic. On the left, there is a vertical bar with a yellow-to-orange gradient. In the center of this bar is a close-up image of a metal key on a textured surface. To the right of the bar, the background is white. The title text is in bold, dark blue, sans-serif font, centered at the top. Below the title is a list of five numbered items. At the bottom left, there is a horizontal orange line above the course information. At the bottom right, there is the university logo and name.

**MACHINE LANGUAGE CODING AND
THE DEBUG SOFTWARE
DEVELOPMENT PROGRAM OF THE PC**

- 4.1 Converting Assembly Language Instructions to Machine Code
- 4.2 Encoding a Complete Program in Machine Code
- 4.3 The PC and Its DEBUG Program
- 4.4 Examining and Modifying the Contents of Memory
- 4.5 Input and Output of Data





THE DEBUG, A SOFTWARE DEVELOPMENT PROGRAM FOR THE PC

- 4.6 Hexadecimal Addition and Subtraction
- 4.7 Loading, Verifying and Saving Machine Language Program
- 4.8 Assembling Instructions with the Assemble Command
- 4.9 Executing Instructions and Programs with the TRACE and GO command
- 4.10 Debugging a Program

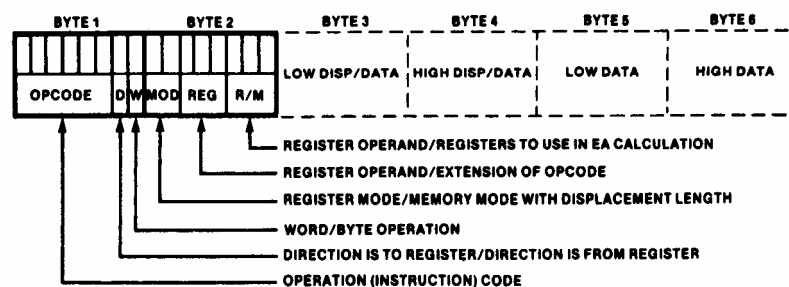
611 37100 微處理機原理與應用 Lecture 04-3

國立台灣大學
生物機電系
林達德



4.1 Converting Assembly Language Instructions to Machine Code

- General instruction format for machine code



611 37100 微處理機原理與應用 Lecture 04-4

國立台灣大學
生物機電系
林達德



4.1 Converting Assembly Language Instructions to Machine Code

■ Byte 1 specification

- ❖ Opcode field (6-bits)
Specifies the operation to be performed
- ❖ Register direction bit (D-bit)
1 – the register operand is a destination operand
0 – the register operand is a source operand
- ❖ Data size bit (W-bit)
1 – 16-bit data size
0 – 8-bit data size

611 37100 微處理機原理與應用 Lecture 04-5

國立台灣大學
生物機電系
林達德



4.1 Converting Assembly Language Instructions to Machine Code

■ Byte 2 specification

- ❖ Mode (MOD) field (2-bits)
Together with R/M field to specify the second operand
- ❖ Register (REG) field (3-bit)
Identifies the register for the first operand
- ❖ Register/Memory (R/M) field (3-bit)

REG	W = 0	W = 1
000	AL	AX
001	CL	CX
010	DL	DX
011	BL	BX
100	AH	SP
101	CH	BP
110	DH	SI
111	BH	DI

Register (REG)
field encoding

611 37100 微處理機原理與應用 Lecture 04-6

國立台灣大學
生物機電系
林達德



4.1 Converting Assembly Language Instructions to Machine Code

Byte 2 specification

CODE	EXPLANATION
00	Memory Mode, no displacement follows*
01	Memory Mode, 8-bit displacement follows
10	Memory Mode, 16-bit displacement follows
11	Register Mode (no displacement)

MOD field and R/M field encoding

*Except when R/M = 110, then 16-bit displacement follows

MOD = 11			EFFECTIVE ADDRESS CALCULATION			
R/M	W = 0	W = 1	R/M	MOD = 00	MOD = 01	MOD = 10
000	AL	AX	000	(BX) + (SI)	(BX) + (SI) + D8	(BX) + (SI) + D16
001	CL	CX	001	(BX) + (DI)	(BX) + (DI) + D8	(BX) + (DI) + D16
010	DL	DX	010	(BP) + (SI)	(BP) + (SI) + D8	(BP) + (SI) + D16
011	BL	BX	011	(BP) + (DI)	(BP) + (DI) + D8	(BP) + (DI) + D16
100	AH	SP	100	(SI)	(SI) + D8	(SI) + D16
101	CH	BP	101	(DI)	(DI) + D8	(DI) + D16
110	DH	SI	110	DIRECT ADDRESS	(BP) + D8	(BP) + D16
111	BH	DI	111	(BX)	(BX) + D8	(BX) + D16

611 37100 微處理機原理與應用 Lecture 04-7

國立台灣大學
生物機電系
林達德



4.1 Converting Assembly Language Instructions to Machine Code

EXAMPLE

`MOV BL, AL`

Encode the above instruction in machine code

Solution:

OPCODE = 100010 (for MOV), D = 0 (source), W = 0 (8-bit)

This leads to BYTE 1 = $10001000_2 = 88_{16}$

In byte 2 the source operand, specified by REG, is AL

REG = 000, MOD = 11, R/M = 011

Therefore, BYTE 2 = $11000011_2 = C3_{16}$

`MOV BL, AL = 88C316`

611 37100 微處理機原理與應用 Lecture 04-8

國立台灣大學
生物機電系
林達德



4.1 Converting Assembly Language Instructions to Machine Code

EXAMPLE

`ADD AX, [SI]`

Encode the above instruction in machine code

Solution:

OPCODE = 000000 (for ADD), D = 1 (dest.), W = 1 (16-bit)

This leads to $\text{BYTE 1} = 00000011_2 = 03_{16}$

In byte 2 the destination operand, specified by REG, is AX

REG = 000, MOD = 00, R/M = 100

Therefore, $\text{BYTE 2} = 00000100_2 = 04_{16}$

`ADD AX, [SI] = 030416`



4.1 Converting Assembly Language Instructions to Machine Code

EXAMPLE

`XOR CL, [1234H]`

Encode the above instruction in machine code

Solution:

OPCODE = 001100 (for XOR), D = 1 (dest.), W = 0 (8-bit)

This leads to $\text{BYTE 1} = 00110010_2 = 32_{16}$

In byte 2 the destination operand, specified by REG, is CL

REG = 001, MOD = 00, R/M = 110

Therefore, $\text{BYTE 2} = 00001110_2 = 0E_{16}$

$\text{BYTE 3} = 34_{16}$ $\text{BYTE 4} = 12_{16}$

`XOR CL, [1234H] = 320E341216`



4.1 Converting Assembly Language Instructions to Machine Code

EXAMPLE

`ADD [BX][DI]+1234H, AX`

Encode the above instruction in machine code

Solution:

OPCODE = 000000 (for ADD), D = 0 (source), W = 1 (16-bit)

This leads to BYTE 1 = $00000001_2 = 01_{16}$

In byte 2 the destination operand, specified by REG, is AX

REG = 000, MOD = 10, R/M = 001

Therefore, BYTE 2 = $10000001_2 = 81_{16}$

BYTE 3 = 34_{16} BYTE 4 = 12_{16}

`ADD [BX][DI]+1234H, AX = 0181341216`



4.1 Converting Assembly Language Instructions to Machine Code

■ Additional one-bit field and their functions

Field	Value	Function
S	0	No sign extension
	1	Sign extend 8-bit immediate data to 16 bits if W=1
V	0	Shift/rotate count is one
	1	Shift/rotate count is specified in CL register
Z	0	Repeat/loop while zero flag is clear
	1	Repeat/loop while zero flag is set

■ Instructions that involve a segment register (SR-field)

Register	SR
ES	00
CS	01
SS	10
DS	11



4.1 Converting Assembly Language Instructions to Machine Code

EXAMPLE

MOV WORD PTR [BP][DI]+1234H, 0ABCDH

Encode the above instruction in machine code

Solution:

This example does not follow the general format

From Fig. 3-1 MOV -> 1100011W, and W = 1 for word-size data

BYTE 1 = 11000111₂ = C7₁₆

BYTE 2 = (MOD)000(R/M) = 10000011₂ = 83₁₆

BYTE 3 = 34₁₆ BYTE 4 = 12₁₆

BYTE 5 = CD₁₆ BYTE 6 = AB₁₆

MOV WORD PTR [BP][DI]+1234H, 0ABCDH = C7833412CDAB₁₆

611 37100 微處理機原理與應用 Lecture 04-13

國立台灣大學
生物機電系
林達德



4.1 Converting Assembly Language Instructions to Machine Code

EXAMPLE

MOV [BP][DI]+1234H, DS

Encode the above instruction in machine code

Solution:

This example does not follow the general format

From Fig. 3-6 MOV -> 10001100, and the instruction is

10001100(MOD)0(SR)(R/M)(DISP)

From Fig. 4-5 we find that for DS, the SR = 11

Therefore, the instruction is coded as

MOV [BP][DI]+1234H, DS
=10001100100110110011010000010010₂
=8C9B3412₁₆

611 37100 微處理機原理與應用 Lecture 04-14

國立台灣大學
生物機電系
林達德



4.2 Encoding a Complete Program in Machine Code

- Steps in encoding a complete assembly program:
 - ❖ Identify the general machine code format (Fig. 3-6)
 - ❖ Evaluate the bit fields (Fig. 4-2,4-3,4-4,4-5)
 - ❖ Express the binary-code instruction in hexadecimal form
- To execute the program, the machine code of the program must be stored in the code segment of memory.
- The first byte of the program is stored at the lowest address.

611 37100 微處理機原理與應用 Lecture 04-15

國立台灣大學
生物機電系
林達德



4.2 Encoding a Complete Program in Machine Code

EXAMPLE

Encode the “block move” program in Fig. 4-6(a) and show how it would be stored in memory starting at address 200₁₆.

Solution:

```
MOV AX, 2000H ;LOAD AX REGISTER
MOV DS, AX ;LOAD DATA SEGMENT ADDRESS
MOV SI, 100H ;LOAD SOURCE BLOCK POINTER
MOV DI, 120H ;LOAD DESTINATION BLOCK POINTER
MOV CX, 10H ;LOAD REPEAT COUNTER
NXTPT: MOV AH, [SI] ;MOVE SOURCE BLOCK ELEMENT TO AH
MOV [DI], AH ;MOVE ELEMENT FROM AH TO DEST. BLOCK
INC SI ;INCREMENT SOURCE BLOCK POINTER
INC DI ;INCREMENT DESTINATION BLOCK POINTER
DEC CX ;DECREMENT REPEAT COUNTER
JNZ NXTPT ;JUMP TO NXTPT IF CX NOT EQUAL TO ZERO
NOP ;NO OPERATION
```

611 37100 微處理機原理與應用 Lecture 04-16

國立台灣大學
生物機電系
林達德



4.2 Encoding a Complete Program in Machine Code

Instruction	Type of instruction	Machine code
MOV AX,200H	Move immediate data to register	1011100000000000010000 ₂ = B8002 ₁₆
MOV DS,AX	Move register to segment register	1000111011011000 ₂ = 8ED8 ₁₆
MOV SI,100H	Move immediate data to register	101111100000000000000001 ₂ = BE0001 ₁₆
MOV DI,120H	Move immediate data to register	101111100100000000000001 ₂ = BF2001 ₁₆
MOV CX,10H	Move immediate data to register	101110010001000000000000 ₂ = B91000 ₁₆
MOV AH,[SI]	Move memory data to register	1000101000100100 ₂ = 8A24 ₁₆
MOV [DI],AH	Move register data to memory	1000100000100101 ₂ = 8825 ₁₆
INC SI	Increment register	01000110 ₂ = 46 ₁₆
INC DI	Increment register	01000111 ₂ = 47 ₁₆
DEC CX	Decrement register	01001001 ₂ = 49 ₁₆
JNZ NXTPT	Jump on not equal to zero	0111010111110111 ₂ = 75F7 ₁₆
NOP	No operation	1001000 ₂ = 90 ₁₆

611 37100 微處理機原理與應用 Lecture 04-17

國立台灣大學
生物機電系
林達德



4.2 Encoding a Complete Program in Machine Code

Memory address	Contents	Instruction
200H	B8H	MOV AX,200H
201H	00H	
202H	20H	
203H	BEH	MOV DS,AX
204H	D8H	
205H	BEH	MOV SI,100H
206H	00H	
207H	01H	
208H	BFH	MOV DI,120H
209H	20H	
20AH	01H	
20BH	B9H	MOV CX,10H
20CH	10H	
20DH	00H	
20EH	8AH	MOV AH,[SI]
20FH	24H	
210H	88H	MOV [DI],AH
211H	25H	
212H	46H	INC SI
213H	47H	INC DI
214H	49H	DEC CX
215H	75H	JNZ \$-9
216H	F7H	
217H	90H	NOP

611 37100 微處理機原理與應用 Lecture 04-18

國立台灣大學
生物機電系
林達德



4.3 The PC and Its DEBUG Program

- Using DEBUG, the programmer can issue commands to the microcomputer.
- Loading the DEBUG program

C:\DEBUG
- Six kinds of information are entered as part of a command:
 - ❖ A command letter
 - ❖ An address
 - ❖ A register name
 - ❖ A file name
 - ❖ A drive name
 - ❖ Data

611 37100 微處理機原理與應用 Lecture 04-19

國立台灣大學
生物機電系
林達德



4.3 The PC and Its DEBUG Program

- The DEBUG program command set

Command	Syntax	Function
Register	R [REGISTER NAME]	Examine or modify the contents of an internal register
Quit	Q	End use of the DEBUG program
Dump	D [ADDRESS]	Dump the contents of memory to the display
Enter	E ADDRESS [LIST]	Examine or modify the contents of memory
Fill	F STARTING ADDRESS ENDING ADDRESS LIST	Fill a block in memory with the data in list
Move	M STARTING ADDRESS ENDING ADDRESS DESTINATION ADDRESS	Move a block of data from a source location in memory to a destination location
Compare	C STARTING ADDRESS ENDING ADDRESS DESTINATION ADDRESS	Compare two blocks of data in memory and display the locations that contain different data
Search	S STARTING ADDRESS ENDING ADDRESS LIST	Search through a block of data in memory and display all locations that match the data in list
Input	I ADDRESS	Read the input port
Output	O ADDRESS, BYTE	Write the byte to the output port
Hex Add/Subtract	H NUM1,NUM2	Generate hexadecimal sum and difference of the two numbers
Unassemble	U [STARTING ADDRESS [ENDING ADDRESS]]	Unassemble the machine code into its equivalent assembler instructions
Name	N FILE NAME	Assign the filename to the data to be written to the disk
Write	W [STARTING ADDRESS [DRIVE STARTING SECTOR NUMBER OF SECTORS]]	Save the contents of memory in a file on a diskette
Load	L [STARTING ADDRESS [DRIVE STARTING SECTOR NUMBER OF SECTORS]]	Load memory with the contents of a file on a diskette
Assemble	A [STARTING ADDRESS]	Assemble the instruction into machine code and store in memory
Trace	T [=ADDRESS] [NUMBER]	Trace the execution of the specified number of instructions
Go	G [=STARTING ADDRESS [BREAKPOINT ADDRESS...]]	Execute the instructions down through the breakpoint address

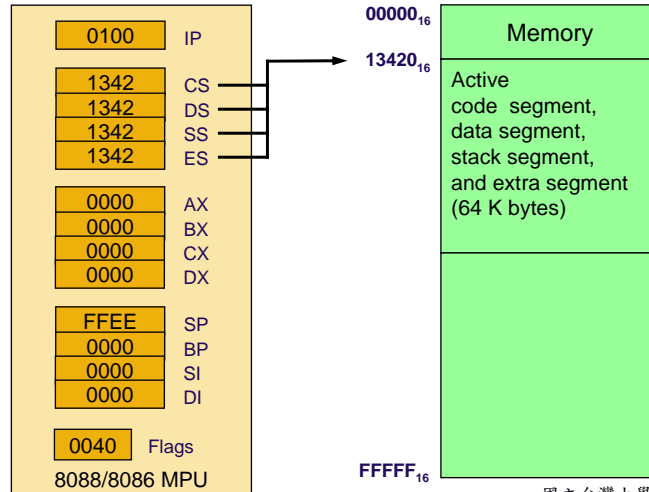
611 37100 微處理機原理與應用 Lecture 04-20

國立台灣大學
生物機電系
林達德



4.3 The PC and Its DEBUG Program

- An initial state when with the loading of DEBUG



4.3 The PC and Its DEBUG Program

- Syntax for the REGISTER (R) command
R [REGISTER NAME]

e.g. -R AX (↵)

AX 0000
:_

:00FF (↵) ;This alter the content of AX
_



4.3 The PC and Its DEBUG Program

EXAMPLE

Verify the initialized state of the 8088 by examining the contents of its registers with the Register command

Solution:

-R (↵)

```
CONSOLE MODE - debug
-R
AX=0000 BX=0000 CX=0000 DX=0000 SP=FFEE BP=0000 SI=0000 DI=0000
DS=0835 ES=0835 SS=0835 CS=0835 IP=0100 NV UP EI PL NZ NA PO NC
0835:0100 E375 JXZ 0177
```



4.3 The PC and Its DEBUG Program

■ Register mnemonics for the R command

Symbol	Register
AX	Accumulator register
BX	Base register
CX	Count register
DX	Data register
SI	Source index register
DI	Destination index register
SP	Stack pointer register
BP	Base pointer register
CS	Code segment register
DS	Data segment register
SS	Stack segment register
ES	Extra segment register
F	Flag register
IP	Instruction pointer



4.3 The PC and Its DEBUG Program

■ Status flag notations

```

CONSOLE MODE - debug
-R
AX=0000 BX=0000 CX=0000 DX=0000 SP=FFEE BP=0000 SI=0000 DI=0000
DS=0B35 ES=0B35 SS=0B35 CS=0B35 IP=0100 NV UP EI PL NZ NA PO NC
0B35:0100 E375 JXZ 0177
  
```

Flag	Meaning	Set	Reset
OF	Overflow	OV	NV
DF	Direction	DN	UP
IF	Interrupt	EI	DI
SF	Sign	NG	PL
ZF	Zero	ZR	NZ
AF	Auxiliary carry	AC	NA
PF	Parity	PE	PO
CF	Carry	CY	NC



4.3 The PC and Its DEBUG Program

EXAMPLE

Issue commands to the DEBUG program on the PC that causes the value in BX to be modified to FF0₁₆ and then verify that this new value is loaded into BX.

Solution:

```

-R BX (↓)
BX 0000
:FF00 (↓)
-R BX (↓)
BX FF00
:_ (↓)
-
  
```



4.3 The PC and Its DEBUG Program

EXAMPLE

Use the Register command to set the parity flag to even parity.
Verify that the flag has been changed.

Solution:

```
-R F (↵)
NV UP EI PL NZ NA PO NC -PE (↵)
-R F (↵)
NV UP EI PL NZ NA PE NC - (↵)
```

611 37100 微處理機原理與應用 Lecture 04-27

國立台灣大學
生物機電系
林達德



4.4 Examining and Modifying the Contents of Memory

■ The commands provided for use in examining and modifying the memory:

- ❖ DUMP
- ❖ ENTER
- ❖ FILL
- ❖ MOVE
- ❖ COMPARE
- ❖ SERACH

611 37100 微處理機原理與應用 Lecture 04-28

國立台灣大學
生物機電系
林達德



4.4 Examining and Modifying the Contents of Memory

■ DUMP Command (D)

The DUMP command allows us to examine the contents of a memory location or a block of consecutive memory location.

D [ADDRESS]

e.g. -D (↵)
-D 1342:100 (↵)
-D DS:100 (↵)
-D 100 (↵)

611 37100 微處理機原理與應用 Lecture 04-29

國立台灣大學
生物機電系
林達德



4.4 Examining and Modifying the Contents of Memory

■ DUMP Command (D)

```
CONSOLE MODE - debug
-D
0B35:0100 E3 75 18 50 A0 D3 96 04-41 E8 2E 01 80 3A E8 29 .u.P...A...:.)
0B35:0110 01 58 89 3E F3 99 C6 06-F5 99 00 E8 34 00 24 08 .X.>...x...4.5
0B35:0120 E8 17 01 AC EB 78 80 3E-B1 98 01 75 03 E8 8D E0 ...x>...u...
0B35:0130 3C 2E 75 09 FE 06 F6 99-C6 06 F5 99 FF 3C 3F 75 <.u...<*u0...<?u
0B35:0140 03 80 CF 02 3C 2A 75 30-80 CF 02 80 3E 2F 9A 00 ...<*u0...>/...
0B35:0150 75 04 E8 24 E8 78 B4 07-80 3E F6 99 00 74 02 B4 u.$x...>...t...
0B35:0160 02 80 3F 2A 26 F5 99 72-EB 86 E1 E3 09 86 E1 E8 .?*&...u!...
0B35:0170 C8 00 86 E1 E2 F7 86 E1-E8 D4 E2 75 21 80 CF 04
```

Address of the first byte of data

ASCII version of the memory data

16 bytes of data per line,
128 bytes per dump

611 37100 微處理機原理與應用 Lecture 04-30

國立台灣大學
生物機電系
林達德



4.4 Examining and Modifying the Contents of Memory

EXAMPLE

Issue a dump command to display the contents of the 32 bytes of memory located at offset 0300₁₆ through 031F₁₆ in the current data segment.

Solution:

-D 300 31F (↵)

```
CONSOLE MODE - debug
-D 300 31F
0B35:0300 59 91 2E C6 07 00 2E 89-1E 68 91 26 88 1D 8D 36 Y.....h.&...6
0B35:0310 6C 91 2E 80 3C 2F 74 36-2E 80 3C 22 74 08 2E F6 |...</t6..<"t...
```



4.4 Examining and Modifying the Contents of Memory

EXAMPLE

Use the Dump command to examine the 16 bytes of memory just below the top of the stack.

Solution:

-D SS:FFEE FFFD (↵)

```
CONSOLE MODE - debug
-R
AX=0000 BX=0000 CX=0000 DX=0000 SP=FFEE BP=0000 SI=0000 DI=0000
DS=0B35 ES=0B35 SS=0B35 CS=0B35 IP=0100 NV UP EI PL NZ NA PO NC
0B35:0100 E375 JCXZ 0177
-D SS:FFEE FFFD
0B35:FFE0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0B35:FFF0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
```



4.4 Examining and Modifying the Contents of Memory

■ ENTER Command (E)

E ADDRESS [LIST]

e.g. -E DS:100 FF FF FF FF FF (↵)

-E DS:100 (↵)

-1342:0100 FF. _ (↵) (Return to end)

-E DS:100 (↵)

-1342:0100 FF. _ (Space bar to continue)

-1342:0100 FF. FF. _

611 37100 微處理機原理與應用 Lecture 04-33

國立台灣大學
生物機電系
林達德



4.4 Examining and Modifying the Contents of Memory

EXAMPLE

Start a data entry sequence by examining the contents of address DS:100 and then, without entering new data, depress the "-" key. What happen?

Solution:

-E DS:100 (↵)

1342:0100 FF. _

Entering "-" causes the display of previous byte storage location.

```
CONSOLE MODE - debug
-E DS:100
0B37:0100 E8. -
0B37:00FF 00. -
0B37:00FE 00. _
```

611 37100 微處理機原理與應用 Lecture 04-34

國立台灣大學
生物機電系
林達德



4.4 Examining and Modifying the Contents of Memory

EXAMPLE

Enter ASCII data to the memory.

Solution:

```
-E DS:200 "ASCII" (↵)
```

```
-E DS:200 'ASCII' (↵)
```

```
CONSOLE MODE - debug
-E DS:200 "ASCII"
-D DS:200 204
0B37:0200 41 53 43 49 49
ASCII
```



4.4 Examining and Modifying the Contents of Memory

■ FILL Command (F)

The FILL command fills a block of consecutive memory locations all with the same data.

F STARTING_ADDRESS ENDING_ADDRESS LIST

e.g. `-F 100 11F 22 (↵)`



4.4 Examining and Modifying the Contents of Memory

EXAMPLE

Initialize all storage locations in the block of memory from DS:120 through DS:13F with the value 33₁₆ and the block of storage locations from DS:140 to DS:15F with the value 44₁₆.

Solution:

```
-F 120 13F 33 (↵)
```

```
-F 140 15F 44 (↵)
```

```
CONSOLE MODE - debug
-
-F 120 13F 33
-F 140 15F 44
-D 120 15F
0B37:0120 33 33 33 33 33 33 33 33-33 33 33 33 33 33 33 33 3333333333333333
0B37:0130 33 33 33 33 33 33 33 33-33 33 33 33 33 33 33 33 3333333333333333
0B37:0140 44 44 44 44 44 44 44 44-44 44 44 44 44 44 44 44 4444444444444444
0B37:0150 44 44 44 44 44 44 44 44-44 44 44 44 44 44 44 44 4444444444444444
```



4.4 Examining and Modifying the Contents of Memory

■ MOVE Command (M)

The MOVE command allows us to copy a block of data from one part of memory to another part.

M START_ADDRESS END_ADDRESS DEST_ADDRESS

e.g. -M 100 11F 200 (↵)



4.4 Examining and Modifying the Contents of Memory

EXAMPLE

Fill each storage location in the block of memory from address DS:100 through DS:11F with the value 11₁₆. Then copy this block of data to a destination block starting at DS:160.

Solution: -F 100 11F 11 (↵)
 -M 100 11F 160 (↵)

```
CONSOLE MODE - debug
-F 100 11F 11
-M 100 11F 160
-D 100 17F
OB37:0100 11 11 11 11 11 11 11 11-11 11 11 11 11 11 11 11 11
OB37:0110 11 11 11 11 11 11 11 11-11 11 11 11 11 11 11 11 11
OB37:0120 33 33 33 33 33 33 33 33-33 33 33 33 33 33 33 33 33
OB37:0130 33 33 33 33 33 33 33 33-33 33 33 33 33 33 33 33 33
OB37:0140 44 44 44 44 44 44 44 44-44 44 44 44 44 44 44 44 44
OB37:0150 44 44 44 44 44 44 44 44-44 44 44 44 44 44 44 44 44
OB37:0160 11 11 11 11 11 11 11 11-11 11 11 11 11 11 11 11 11
OB37:0170 11 11 11 11 11 11 11 11-11 11 11 11 11 11 11 11 11
```



4.4 Examining and Modifying the Contents of Memory

■ COMPARE Command (C)

The COMPARE command allows us to compare the contents of two blocks of data to determine if they are or are not the same.

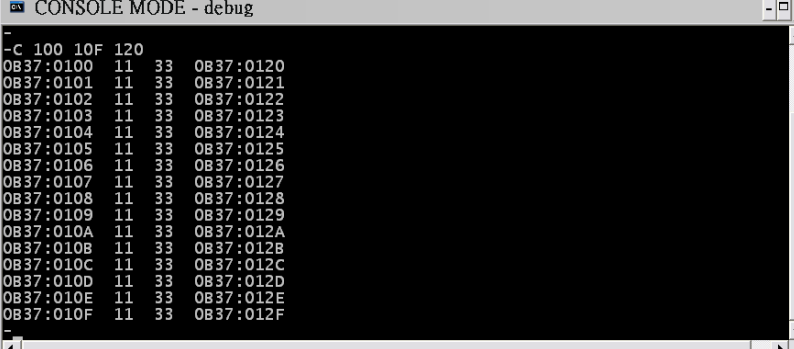
C START_ADDRESS END_ADDRESS DEST_ADDRESS

e.g. -C 100 10F 120 (↵)



4.4 Examining and Modifying the Contents of Memory

■ COMPARE Command (C)



```
CONSOLE MODE - debug
-C 100 10F 120
0B37:0100 11 33 0B37:0120
0B37:0101 11 33 0B37:0121
0B37:0102 11 33 0B37:0122
0B37:0103 11 33 0B37:0123
0B37:0104 11 33 0B37:0124
0B37:0105 11 33 0B37:0125
0B37:0106 11 33 0B37:0126
0B37:0107 11 33 0B37:0127
0B37:0108 11 33 0B37:0128
0B37:0109 11 33 0B37:0129
0B37:010A 11 33 0B37:012A
0B37:010B 11 33 0B37:012B
0B37:010C 11 33 0B37:012C
0B37:010D 11 33 0B37:012D
0B37:010E 11 33 0B37:012E
0B37:010F 11 33 0B37:012F
```

Results produced when unequal data are found with a COMPARE command



4.4 Examining and Modifying the Contents of Memory

■ SEARCH Command (S)

The SEARCH command can be used to scan through a block of data in memory to determine whether or not it contains specific data.

C START_ADDRESS END_ADDRESS LIST

e.g. -S 100 17F 33 (↵)





4.5 Input and Output of Data

■ OUTPUT Command (O)

The OUTPUT command write data to an input port of the 64K byte-wide ports of 8088 I/O.

O ADDRESS BYTE

e.g. -O 61 4F (↵)

This command causes the value $4F_{16}$ to be written into the byte-wide output port at address 0061_{16}



4.6 Hexadecimal Addition and Subtraction

■ HEXADECIMAL Command (H)

The HEXADECIMAL command provides the ability to add and subtract hexadecimal numbers.

H NUM1 NUM2

e.g. -H ABC0 0FFF (↵)
BBBF 9BC1

-H BBBF A (↵)
BBC9 BBB5

*Both number and results are limited to four hexadecimal digits.



4.6 Hexadecimal Addition and Subtraction

EXAMPLE

Use the H command to find the negative of the number 0009_{16} .

Solution:

```
-H 0 9 (-)
0009 FFF7
```

$FFF7_{16}$ is the negative of 9_{16} expressed in 2's complement form.



4.6 Hexadecimal Addition and Subtraction

EXAMPLE

If a byte of data is located at physical address $02A34_{16}$ and the data segment register contains 0150_{16} , what value must be loaded into the source index register such that DS:SI points to the byte storage location?

Solution:

```
-H 2A34 1500 (-)
3F34 1534
```

This shows that SI must be loaded with the value 1534_{16} .





4.7 Loading, Verifying and Saving Machine Language Program

- An example to load an instruction

MOV BL, AL

The machine code is $88C3_{16}$

-E CS:100 88 C3 (↵)

-D CS:100 101 (↵)

1342:0100 88 C3

611 37100 微處理機原理與應用 Lecture 04-49

國立台灣大學
生物機電系
林達德



4.7 Loading, Verifying and Saving Machine Language Program

- UNASSEMBLE Command (U)

The UNASSEMBLE command converts machine code instructions to their equivalent assembly language source statement.

U [STARTING_ADDRESS [ENDING_ADDRESS]]

e.g. -U CS:100 101 (↵)
 1342:0100 88C3 MOV BL, AL

611 37100 微處理機原理與應用 Lecture 04-50

國立台灣大學
生物機電系
林達德



4.7 Loading, Verifying and Saving Machine Language Program

EXAMPLE

Use a sequence of commands to load, verify loading, and unassemble the machine code instruction 0304H. Load the instruction at address CS:200.

Solution: -E CS:200 03 04 (↵)
 -D CS:200 201 (↵)
 -U CS:200 201 (↵)
 ADD AX, [SI]

```
CONSOLE MODE - debug
- E CS:200 03 04
- D CS:200 201
0B37:0200 03 04
- U CS:200 201
0B37:0200 0304
ADD AX, [SI]
```



4.7 Loading, Verifying and Saving Machine Language Program

■ WRITE Command (W)

The WRITE command gives the ability to save data stored in memory on a diskette.

W [START_ADDRESS [DRIVE START_SECTOR NUM_SECTOR]]

e.g. -W CS:200 1 10 1 (↵)
 -W 200 1 10 1 (↵)

Drive B

1 Sector = 512 Byte

* Be caution in saving program in a disk, especially the hard drive.





4.7 Loading, Verifying and Saving Machine Language Program

■ LOAD Command (L)

The LOAD command gives the ability to reload memory from a diskette.

L [START_ADDRESS [DRIVE START_SECTOR NUM_SECTOR]]

e.g. -L CS:300 1 10 1 (↵)

The reloading of the instruction can be verified by U command

e.g. -U CS:300 301 (↵)
 1342:300 301 ADD AX, [SI]

611 37100 微處理機原理與應用 Lecture 04-53

國立台灣大學
生物機電系
林達德



4.7 Loading, Verifying and Saving Machine Language Program

EXAMPLE

Enter the machine code of the block move program. The program is to be loaded into memory starting at address CS:100. Verify, unassemble, and save the code.

Solution:

-E CS:100 B8 00 20 8E D8 BE 00 01 BF 20 01 B9 10
00 8A 24 88 25 46 (↵)

-D CS:100 117(↵)

-U CS:100 117(↵)

-W CS:100 1 100 1 (↵)

611 37100 微處理機原理與應用 Lecture 04-54

國立台灣大學
生物機電系
林達德



4.7 Loading, Verifying and Saving Machine Language Program

```
CONSOLE MODE - debug
-E CS:100 B8 00 20 8E D8 BE 00 01 BF 20 01 B9 10 00 8A 24
-E CS:110 88 25 46 47 49 75 F7 90
-D CS:100 117
0B37:0100 B8 00 20 8E D8 BE 00 01-BF 20 01 B9 10 00 8A 24  ....$
0B37:0110 88 25 46 47 49 75 F7 90  .%FGIU..
-U CS:100 117
0B37:0100 B80020      MOV    AX,2000
0B37:0103 8ED8      MOV    DS,AX
0B37:0105 BE0001      MOV    SI,0100
0B37:0108 BF2001      MOV    DI,0120
0B37:010B B91000      MOV    CX,0010
0B37:010E 8A24      MOV    AH,[SI]
0B37:0110 8825      MOV    [DI],AH
0B37:0112 46      INC    SI
0B37:0113 47      INC    DI
0B37:0114 49      DEC    CX
0B37:0115 75F7      JNZ   010E
0B37:0117 90      NOP
-W CS:100 0 100 1
```



4.7 Loading, Verifying and Saving Machine Language Program

NAME Command (N)

The NAME command, along with the WRITE command, gives the ability to save a program on the diskette under a file name.

N FILE NAME

- ❖ The BX, CX registers must be updated to identify the size of the program that is to be saved in the file.
- ❖ After BX, CX registers have been initialized, the write command is used to save the program.
- ❖ To reload the program, the command sequence is

N FILE NAME

L [STARTING ADDRESS]



4.7 Loading, Verifying and Saving Machine Language Program

EXAMPLE

Save a machine code program into a file.

Solution:

```
-N A:BLK.1 (↵) ; Give a file name in disk A
-R CX (↵) ; Give a program size of 1816 bytes
CX XXXX
:18
-R BX (↵)
BX XXXX
:0 (↵)
W CS:100 (↵) ; Save the program in disk A
```

611 37100 微處理機原理與應用 Lecture 04-57

國立台灣大學
生物機電系
林達德



4.7 Loading, Verifying and Saving Machine Language Program

EXAMPLE

Reload a program into memory.

Solution:

```
-N A:BLK.1 (↵) ; Give a file name in disk A
-L CS:100 (↵) ; Load the program name BLK.1 in disk A
```

```
C:\DOS>REN A:BLK.1 BLK.EXE (↵) ; Rename the file
```

```
C:\DOS>DEBUG A:BLK.EXE (↵) ; Load the program directly
```

```
C:\DOS>A:BLK.EXE (↵) ; Run the program
```

611 37100 微處理機原理與應用 Lecture 04-58

國立台灣大學
生物機電系
林達德



4.8 Assembling Instructions with the Assemble Command

■ ASSEMBLE Command (A)

The ASSEMBLE command let us automatically assemble the instructions of a program.

A [STARTING_ADDRESS]

e.g. -A CS:100 (↵)
 1342:0100 _
 1342:0100 ADD [BX+SI+1234], AX (↵)
 1342:0104 _
 -D CS:100 103 (↵)



4.8 Assembling Instructions with the Assemble Command

EXAMPLE

Assemble a complete program with the ASSEMBLE command.

Solution:

```
-A CS:200 (↵)
0B35:0200 MOV AX, 2000 (↵)
0B35:0203 MOV DS, AX (↵)
0B35:0205 MOV SI, 100 (↵)
. . . .
. . . .
0B35:0217 NOP (↵)
0B35:0218 (↵)
```



4.8 Assembling Instructions with the Assemble Command

- Assemble a program with ASSEMBLE command

```
CONSOLE MODE - DEBUG
-A CS:200
OB35:0200 MOV AX,2000
OB35:0203 MOV DS,AX
OB35:0205 MOV SI,100
OB35:0208 MOV DI,120
OB35:020B MOV CX,10
OB35:020E MOV AH,[SI]
OB35:0210 MOV [DI],AH
OB35:0212 INC SI
OB35:0213 INC DI
OB35:0214 DEC CX
OB35:0215 JNZ 20E
OB35:0217 NOP
OB35:0218
```



4.8 Assembling Instructions with the Assemble Command

- Unassemble a program with UNASSEMBLE command

```
CONSOLE MODE - DEBUG
-U CS:200 217
OB35:0200 B80020 MOV AX,2000
OB35:0203 8ED8 MOV DS,AX
OB35:0205 BE0001 MOV SI,0100
OB35:0208 BF2001 MOV DI,0120
OB35:020B B91000 MOV CX,0010
OB35:020E 8A24 MOV AH,[SI]
OB35:0210 8825 MOV [DI],AH
OB35:0212 46 INC SI
OB35:0213 47 INC DI
OB35:0214 49 DEC CX
OB35:0215 75F7 JNZ 020E
OB35:0217 90 NOP
```





4.9 Executing Instructions and Programs with the TRACE and GO command

■ TRACE Command (T)

The TRACE command provides the programmer with the ability to execute the program one instruction at a time.

T [=STARTING_ADDRESS] [NUMBER]

e.g. -T =CS:100 (↵)

-T (↵)

-T =CS:100 3 (↵)

611 37100 微處理機原理與應用 Lecture 04-63

國立台灣大學
生物機電系
林達德



4.9 Executing Instructions and Programs with the TRACE and GO command

EXAMPLE

Load and trace a program.

Solution:

-L CS:100 1 10 1 (↵)

-U 100 101 (↵)

-R AX (↵)

AX 0000

:1111 (↵)

-R SI (↵)

SI 0000

:1234 (↵)

-E DS:1234 22 22 (↵)

-T =CS:100 (↵)

611 37100 微處理機原理與應用 Lecture 04-64

國立台灣大學
生物機電系
林達德



4.9 Executing Instructions and Programs with the TRACE and GO command

```
CONSOLE MODE - DEBUG
-L CS:100 0 10 1
-U 100 101
OB35:0100 0304      ADD     AX,[SI]
-R AX
AX 0000
:1111
-R SI
SI 0000
:1234
-E DS:1234 22 22
-R
AX=1111 BX=0000 CX=0000 DX=0000 SP=FFEE BP=0000 SI=1234 DI=0000
DS=0B35 ES=0B35 SS=0B35 CS=0B35 IP=0100 NV UP EI PL NZ NA PO NC
OB35:0100 0304      ADD     AX,[SI]          DS:1234=22
-D DS:1234 1235
OB35:1230          22 22          ""
-T =CS:100

AX=3333 BX=0000 CX=0000 DX=0000 SP=FFEE BP=0000 SI=1234 DI=0000
DS=0B35 ES=0B35 SS=0B35 CS=0B35 IP=0102 NV UP EI PL NZ NA PE NC
OB35:0102 0000      ADD     [BX+SI],AL      DS:1234=22
```



4.9 Executing Instructions and Programs with the TRACE and GO command

■ GO Command (G)

The GO command is typically used to run programs that are already working or to execute programs in the later stages or debugging.

G [=STARTING_ADDRESS [BREAKPOINT_ADDRESS_LIST]]

e.g. -G =CS:200 217 (↵)

 -G =CS:100 (↵)

 -G (↵)



4.9 Executing Instructions and Programs with the TRACE and GO command

EXAMPLE

Use GO command to execute a program and examine the result.

Solution:

```
-N A:BLK.EXE (↵) ; Define the program file to be loaded
-L CS:200 (↵) ; Load the program at CS:200
-R DS (↵)
DS 1342
:2000 (↵) ; Define the data segment address
-F DS:100 10F FF (↵) ; Fill memory with FF
-F DS:120 12F 00 (↵) ; Fill memory with 00
-R DS (↵)
DS 2000
:1342 ; Store data segment with 134216
```

611 37100 微處理機原理與應用 Lecture 04-67

國立台灣大學
生物機電系
林達德



4.9 Executing Instructions and Programs with the TRACE and GO command

EXAMPLE

Use GO command to execute a program and examine the result.

Solution: (continued)

```
-R (↵) ; Show data register status
-U CS:200 217 (↵) ; Unassemble the program
-G =CS:200 20E (↵) ; Execute the program to CS:20E
-G =CS:20E 215 (↵) ; Execute the program to CS:215
-D DS:100 10F (↵) ; Display memory at DS:100
-D DS:120 12F (↵) ; Display memory at DS:120
-G =CS:215 217 (↵) ; Execute the program to CS:217
-D DS:100 10F (↵) ; Display memory at DS:100
-D DS:120 12F (↵) ; Display memory at DS:120
```

611 37100 微處理機原理與應用 Lecture 04-68

國立台灣大學
生物機電系
林達德



4.9 Executing Instructions and Programs with the TRACE and GO command

```
CONSOLE MODE - DEBUG
-D DS:100 10F
2000:0100 FF FF FF FF FF FF FF-FF FF FF FF FF FF FF FF .....
-D DS:120 12F
2000:0120 FF 00 00 00 00 00 00-00 00 00 00 00 00 00 .....
-G =CS:115 117

AX=FF00 BX=FFFF CX=0000 DX=0000 SP=FFFE BP=0000 SI=0110 DI=0130
DS=2000 ES=0B99 SS=0B99 CS=0B99 IP=0117 NV UP EI PL ZR NA PE NC
0B99:0117 90          NOP
-D DS:100 10F
2000:0100 FF FF FF FF FF FF FF-FF FF FF FF FF FF FF FF .....
-D DS:120 12F
2000:0120 FF FF FF FF FF FF FF-FF FF FF FF FF FF FF FF .....
-
```



4.10 Debugging a Program

- Errors in a program are also referred to as **bugs**; the process of removing them is called **debugging**.
- Two types of errors
 - ❖ Syntax error
 - ❖ Execution error
- A syntax error is an error caused by not following the rules for coding or entering an instruction. These types of errors are typically identified by the microcomputer and signaled to user with an error message
- In the DEBUG environment, the TRACE command is usually used to debug execution errors.



4.10 Debugging a Program

■ Review of the DEBUG commands

Command	Syntax	Function
Register	R [REGISTER NAME]	Examine or modify the contents of an internal register
Quit	Q	End use of the DEBUG program
Dump	D [ADDRESS]	Dump the contents of memory to the display
Enter	E ADDRESS [LIST]	Examine or modify the contents of memory
Fill	F STARTING ADDRESS ENDING ADDRESS LIST	Fill a block in memory with the data in list
Move	M STARTING ADDRESS ENDING ADDRESS DESTINATION ADDRESS	Move a block of data from a source location in memory to a destination location
Compare	C STARTING ADDRESS ENDING ADDRESS DESTINATION ADDRESS	Compare two blocks of data in memory and display the locations that contain different data
Search	S STARTING ADDRESS ENDING ADDRESS LIST	Search through a block of data in memory and display all locations that match the data in list
Input	I ADDRESS	Read the input port
Output	O ADDRESS, BYTE	Write the byte to the output port
Hex Add/Subtract	H NUM1,NUM2	Generate hexadecimal sum and difference of the two numbers
Unassemble	U [STARTING ADDRESS [ENDING ADDRESS]]	Unassemble the machine code into its equivalent assembler instructions
Name	N FILE NAME	Assign the filename to the data to be written to the disk
Write	W [STARTING ADDRESS [DRIVE STARTING SECTOR NUMBER OF SECTORS]]	Save the contents of memory in a file on a diskette
Load	L [STARTING ADDRESS [DRIVE STARTING SECTOR NUMBER OF SECTORS]]	Load memory with the contents of a file on a diskette
Assemble	A [STARTING ADDRESS]	Assemble the instruction into machine code and store in memory
Trace	T [=ADDRESS] [NUMBER]	Trace the execution of the specified number of instructions
Go	G [=STARTING ADDRESS [BREAKPOINT ADDRESS...]]	Execute the instructions down through the breakpoint address

