

A thick black L-shaped frame is positioned on the left and bottom edges of the slide, framing the main title and chapter information.

BACS1024

INTRODUCTION TO

COMPUTER SYSTEMS

Chapter 7: Assembly Language Fundamental – Part II

0. Overview

1. Unconditional Jump & Loop Instructions
2. Conditional Jump & Loop Instructions
3. Video & Keyboard Processing
4. BIOS Interrupts

1. Unconditional Jump & Loop Instructions

1. Unconditional Jump & Loop Instructions

■ **JMP** Instruction

- ☐ Transfers control under all circumstances
- ☐ Allow transfer of control to the target address (any instruction that has labelled)
- ☐ Format: **JMP short / near / far address**

Distance	Short	Near	Far
Instructions	-128 to 127	-32k to +32K	<-32K or >32K
	Same segment	Same segment	Another segment
JMP	✓	✓	✓
Jnnn	✓	80386 / 486	X
LOOP	✓	X	X

1. Unconditional Jump & Loop Instructions

■ **JMP** Instruction

- ❑ The **JMP** and **LOOP** instructions require an operand that refers to the label of an instruction
- ❑ E.g.:

JMP A10

...

A10: ...

Backward Jump	Forward Jump
A20: ... JMP A20	JMP A30 A30: ...

1. Unconditional Jump & Loop Instructions

■ **LOOP** Instruction

- ❑ Repeat a block of statements for a specific number.
- ❑ The **CX** is automatically used as a counter & is decremented each time the **LOOP** repeats
- ❑ Format: **LOOP shortAddress**
- ❑ E.g.: Increase the value 1, 2, 3, 4, 5

```
        MOV AX,00H          ;AX = 0
        MOV CX,05           ;CX = 5

L1:
        INC AX              ;AX = 1, 2, 3, 4, 5
        LOOP L1             ;CX = 4, 3, 2, 1, 0
```

1. Unconditional Jump & Loop Instructions

■ **LOOP** Instruction

- E.g.: Calculates the sum of the integers 1+2+3+4+5

```
MOV AX,00H      ;AX = 0
```

```
MOV BX,00H      ;BX = 0
```

```
MOV CX,05       ;CX = 5
```

```
L1:
```

```
INC AX          ;AX = 1, 2, 3, 4, 5
```

```
ADD BX,AX       ;BX = 1, 3, 5, 9, 14
```

```
LOOP L1         ;Decrement CX, repeat if nonzero
```

```
                ;CX = 4, 3, 2, 1, 0
```

1. Unconditional Jump & Loop Instructions

■ **LOOP** Instruction

- ❑ E.g.: Calculates the sum of an array of 8-bit integer

.DATA

```
arraySum    DB 10h, 20H, 30H, 40H
```

.CODE

```
MOV DI, OFFSET arraySum    ;address
MOV CX, 4                  ;set loop counter
MOV AX, 0                  ;zero the accumulator
```

L1:

```
ADD AL, [DI]              ;add the integer
INC DI                    ;point to next
LOOP L1
```


1. Unconditional Jump & Loop Instructions

■ Nested LOOP Instruction

- ❑ When creating a **LOOP** inside another **LOOP**, special consideration must be given to the outer loop counter in **CX**

.DATA

TEMP DW ?

.CODE

MOV CX, 100

L1:

MOV TEMP, CX ;save outer loop count

MOV CX, 20 ;set inner loop count

L1:

...

LOOP L2 ;repeat the inner loop

MOV CX, TEMP ;restore outer loop count

LOOP L1 ;repeat outer loop

2. Conditional Jump & Loop Instructions

2. Conditional Jump & Loop Instructions

- **CMP Instruction**

- ☐ Used to compare two numeric data values
- ☐ Perform an implied subtraction of a source operand from a destination operand
- ☐ Affect the **AF, CF, SF, PF** and **ZF** flags
- ☐ Format: **CMP destination, source**

2. Conditional Jump & Loop Instructions

- **CMP Instruction**

- ☐ Compare between two unsigned operands

CMP Result	ZF	CF
Destination < Source	0	1
Destination > Source	0	0
Destination = Source	1	0

- ☐ Compare between two signed operands

CMP Result	Flags
Destination < Source	SF ≠ OF
Destination > Source	SF ≠ OF
Destination = Source	ZF = 1

2. Conditional Jump & Loop Instructions

- **CMP** Instruction

□ E.g.:

Destination < Source	Destination > Source	Destination = Source
<code>MOV AX, 5</code> <code>CMP AX, 10</code>	<code>MOV SI, 105</code> <code>CMP SI, 0</code>	<code>MOV AX, 1000</code> <code>MOV CX, 1000</code> <code>CMP CX, AX</code>
ZF = 0	ZF = 0	ZF = 1
CF = 1	CF = 0	CF = 0

2. Conditional Jump & Loop Instructions

- **Conditional Structure**

- ❑ Logic structure can be implemented using a combination of comparisons and jumps
- ❑ 2 steps:
 1. An operation such as **CMP** or **AND** modifies the CPU status flags
 2. A conditional jump instruction tests the flags and causes a branch to new address
- ❑ E.g.: **JZ** instruction jumps to Label **L1** if **ZF** was set

```
CMP AX, 0
```

```
JZ L1      ; Jump if ZF = 1
```

2. Conditional Jump & Loop Instructions

- **Jnnn Instruction**

- ❑ A conditional jump instruction branches to a destination label when a status flag condition is true
- ❑ Otherwise, the instruction immediately following the conditional jump is executed
- ❑ Format: **Jnnn destination**
- ❑ 4 groups of conditional jump instructions:
 1. Jumps based on specific flag values
 2. Jumps based on equality between operands or the value of CX
 3. Jumps based on comparisons of unsigned operands.
 4. Jumps based on comparisons of signed operands.

2. Conditional Jump & Loop Instructions

- **Jnnn** Instruction

1. Jumps based on specific flag values

Mnemonic	Description	Flags / Registers
JZ	Jump if zero	ZF = 1
JNZ	Jump if not zero	ZF = 0
JC	Jump if carry	CF = 1
JNC	Jump if not carry	CF = 0
JO	Jump if overflow	OF = 1
JNO	Jump if not overflow	OF = 0
JS	Jump if signed	SF = 1
JNS	Jump if not signed	SF = 0
JP	Jump if parity (even)	PF = 1
JNP	Jump if not parity (odd)	PF = 0

2. Conditional Jump & Loop Instructions

- **Jnnn** Instruction

1. Jumps based on specific flag values

❖ E.g.:

```
MOV AL, 85H
MOV BL, 8AH
AND AL, BL
JO L1           ; Jump if OF = 1

L1:
...
```

2. Conditional Jump & Loop Instructions

- **Jnnn** Instruction

2. Jumps based on equality between operands or the value of CX

Mnemonic	Description
JE	Jump if equal
JNE	Jump if not equal
JCXZ	Jump if CX = 0

2. Conditional Jump & Loop Instructions

- **Jnnn** Instruction

2. Jumps based on equality between operands or the value of CX

❖ E.g.:

```
MOV DL, B0H
AND DL, 10110000B
JE L2           ;Jump if equal
JNE L3          ;Jump if not equal
L2:  ...
L3:  ...
```

2. Conditional Jump & Loop Instructions

- **Jnnn** Instruction

3. Jumps based on comparisons of unsigned operands.

Mnemonic	Description
JA	Jump if above (if <i>destination > source</i>)
JNBE	Jump if not below or equal
JAE	Jump if above or equal (if <i>destination ≥ source</i>)
JNB	Jump if not below
JB	Jump if below (if <i>destination < source</i>)
JNAE	Jump if not above or equal
JBE	Jump if below or equal (if <i>destination ≤ source</i>)
JNA	Jump if not above

2. Conditional Jump & Loop Instructions

- **Jnnn** Instruction

3. Jumps based on comparisons of unsigned operands.

❖ E.g.:

```
MOV DL, +127
CMP AL, -128
JA L2           ; Jump >
JB L3           ; Jump <
L2:  ...
L3:  ...
```

2. Conditional Jump & Loop Instructions

- **Jnnn** Instruction

4. Jumps based on comparisons of signed operands.

Mnemonic	Description
JG	Jump if greater (if <i>destination > source</i>)
JNLE	Jump if not less than or equal
JGE	Jump if greater than or equal (if <i>destination \geq source</i>)
JNL	Jump if not less
JL	Jump if less (if <i>destination < source</i>)
JNGE	Jump if not greater than or equal
JLE	Jump if less than or equal (if <i>destination \leq source</i>)
JNG	Jump if not greater

2. Conditional Jump & Loop Instructions

- **Jnnn** Instruction

4. Jumps based on comparisons of signed operands.

❖ E.g.:

```
MOV DL, 127
CMP AL, 128
JG L2           ; Jump >
JL L3           ; Jump <
L2:  ...
L3:  ...
```

2. Conditional Jump & Loop Instructions

- **LOOPZ & LOOPE Instruction**

- ☐ Loop if zero and loop if equal
- ☐ Continue looping as long as CX is zero / zero condition is set
- ☐ Useful when scanning an array for the first element that does not match a given value

- **LOOPNZ & LOOPNE Instruction**

- ☐ Loop if not zero and loop if not equal
- ☐ Continue looping as long as CX is not zero / zero condition is not set
- ☐ Useful when scanning an array for the first element that match a given value

3. Video & Keyboard Processing

3. Video & Keyboard Processing

■ Screen Features

- Typical computer screens are features with 25 rows and 80 columns

Screen location	Decimal format		Hex format	
	Row	Column	Row	Column
Upper left corner	00	00	00h	00h
Upper right corner	00	79	00h	4fh
Center	12	39/40	0ch	27h/28h
Lower left corner	24	00	18h	00h
Lower right corner	24	79	18h	4fh

4. BIOS Interrupts

4. BIOS Interrupts

■ **INT** instruction

- ❑ The **INT** (Interrupt) instruction handles BIOS operations.
- ❑ Lower level interrupts (**INT 10H** function)
 - ❖ Go directly to BIOS
 - ❖ For screen handling
- ❑ Higher level interrupt (**INT 21H** function)
 - ❖ Go to the operating systems
 - ❖ For displaying output & accepting keyboard input

4. BIOS Interrupts

■ INT instruction

- ❑ Function code specifies the action to be taken
- ❑ Insert a function code in **AH** register to identify the type of service the interrupt is to be performed.

INT 10H		INT 21H	
02H	Set cursor	01H	Input byte
06H	Scroll screen	02H	Output byte
		07H	Input byte (no echo)
		09H	Output string
		0AH	Input string

4. BIOS Interrupts

■ INT 10H instruction & Function 02H

- ☐ Set cursor's position determines where the next character is to be displayed / entered
- ☐ E.g.:

```
MOV AH, 02H      ; set function (cursor)
MOV BH, 00       ; set page number 0 (0 = current page)
MOV DH, 08       ; set row number
MOV DL, 15       ; set column number
INT 10H          ; call interrupt service
```

4. BIOS Interrupts

■ INT 10H instruction & Function 06H

- ❑ Clear all / part of a display beginning at any screen location & ending at any higher numbered location on the screen
- ❑ E.g.:

```
MOV AH, 0600H    ; AH = 06 (scroll)
                  ; AL = 00 (full screen)
MOV BH, 71H      ; background = white (7)
                  ; foreground = blue (1)
MOV CX, 0000H    ; upper left row:column
MOV DX, 1847H    ; lower right row:column
INT 10H          ; call interrupt service
```

4. BIOS Interrupts

■ INT 10H instruction & Function 06H

□ Color code

❖ **Background: 8** colors & **Foreground: 16** color

COLOR	I	R	G	B	HEX	COLOR	I	R	G	B	HEX
Black	0	0	0	0	0	Gray	1	0	0	0	8
Blue	0	0	0	1	1	Light Blue	1	0	0	1	9
Green	0	0	1	0	2	Light green	1	0	1	0	A
Cyan	0	0	1	1	3	Light Cyan	1	0	1	1	B
Red	0	1	0	0	4	Light red	1	1	0	0	C
Magenta	0	1	0	1	5	Light Magenta	1	1	0	1	D
Brown	0	1	1	0	6	Yellow	1	1	1	0	E
White	0	1	1	1	7	Bright white	1	1	1	1	F

4. BIOS Interrupts

■ **INT 21H** instruction & Function **01H**

- ☐ Input byte
- ☐ Accept single byte from keyboard
- ☐ User input byte is stored in **AL** register
- ☐ E.g.:

```
MOV AH, 01H      ; request keyboard input
INT 21H          ; call interrupt service
```

4. BIOS Interrupts

■ **INT 21H** instruction & Function **02H**

- ☐ Output byte
- ☐ Load the byte to be displayed in **DL** register
- ☐ E.g.:

```
MOV AH, 02H      ; request output byte
MOV DL, 61H      ; byte to be display
                  ; 61H = ASCII 'a'
INT 21H          ; call interrupt service
```

4. BIOS Interrupts

■ **INT 21H** instruction & Function **07H**

- ☐ Input byte
- ☐ Accept single byte from keyboard but no echoed on screen
- ☐ Could be used to key in password that is to be invisible
- ☐ User input byte is stored in **AL** register
- ☐ E.g.:

```
MOV AH, 07H      ; request keyboard input
INT 21H          ; call interrupt service
```

4. BIOS Interrupts

■ INT 21H instruction & Function 09H

- ❑ Output string
- ❑ Requires definition of a display string in data are, immediately followed by a dollar sign ('\$' or 24H) as a delimiter to mark the end of the string
- ❑ Uses **LEA** instruction to load the effective start address of the string
- ❑ E.g.:

.DATA

```
str DB 'Hello', '$'  
str1 DB 'Hello$'
```

.CODE

```
MOV AH, 09H  
LEA DX, str1  
INT 21H
```

4. BIOS Interrupts

■ INT 21H instruction & Function 02H

- ❑ New line
- ❑ To display cursor on new line, use Carriage Return (CR) character - 0DH and Line Feed (LF) character - 0AH
- ❑ E.g.:

```
MOV AH,02H      ;for byte output
MOV DL,0DH      ;CR
INT 21H         ;call interrupt service
MOV DL, 0AH     ;LF
INT 21H         ;call interrupt service
```

4. BIOS Interrupts

■ INT 21H instruction & Function 0AH

- ☐ Input string
- ☐ The input area for keyed-in byte requires a **parameter** list containing specified fields that the **INT** operation is to process
- ☐ E.g.:

.DATA

	ARRAY	LABEL	BYTE		;name
	MAX	DB	20		;max no. of input
	ACT	DB	?		;actual no. of input
string	ARRAYDATA	DB	20 DUP (0)		;data are for input

.CODE

MOV AH, 0AH	;request keyboard input
LEA DX, ARRAY	;load address of parameter
INT 21H	;call interrupt service

4. BIOS Interrupts

■ INT 21H instruction & Function 0AH

- The **LABEL** directive tells the assembler to align on a **BYTE** boundary and give the location the name **ARRAY**

MAX ACT < ----- ARRAYDATA ----- >



4. BIOS Interrupts

■ INT 21H instruction & Function 0AH

- ☐ Key in the name **Wilson+<enter>**
- ☐ The parameter list appears as:

ASCII	20	6	W	i	l	s	o	n	
Hex	14	06	57	69	6C	73	6F	6E	

- ☐ This operation accepts and acts on **<Backspace>** character, but does not add it to the count
- ☐ This operation bypasses extended function keys such as **F1, Home, PgUp** and Arrows. If the mentioned function keys are expected, user INT **16H** or INT **21H** with function **01H**

Chapter Review

Chapter Review

1. Unconditional Jump & Loop Instructions

- ❑ **JMP** instruction
- ❑ **LOOP** instruction

2. Conditional Jump & Loop Instructions

- ❑ **CMP** instruction
- ❑ **Jnnn** instructions
- ❑ **LOOPnn** instructions

3. Video & Keyboard Processing

- ❑ Screen features

4. BIOS Interrupts

- ❑ **INT 10H**
 - ❖ **02H**
 - ❖ **06H**
- ❑ **INT 21H**
 - ❖ **01H**
 - ❖ **02H**
 - ❖ **07H**
 - ❖ **09H**
 - ❖ **0AH**