CSC 222: Computer Organization & Assembly Language

6 – Interrupt Driven IO

Outline

- Interrupts
- Input Output Instructions
- Sample Programs

References

- ***Chapter 3, 4,*** Ytha Yu and Charles Marut, "Assembly Language Programming and Organization of IBM PC"
- ***Chapter 3***, Assembly Language for Intel Based-Computers

Interrupts

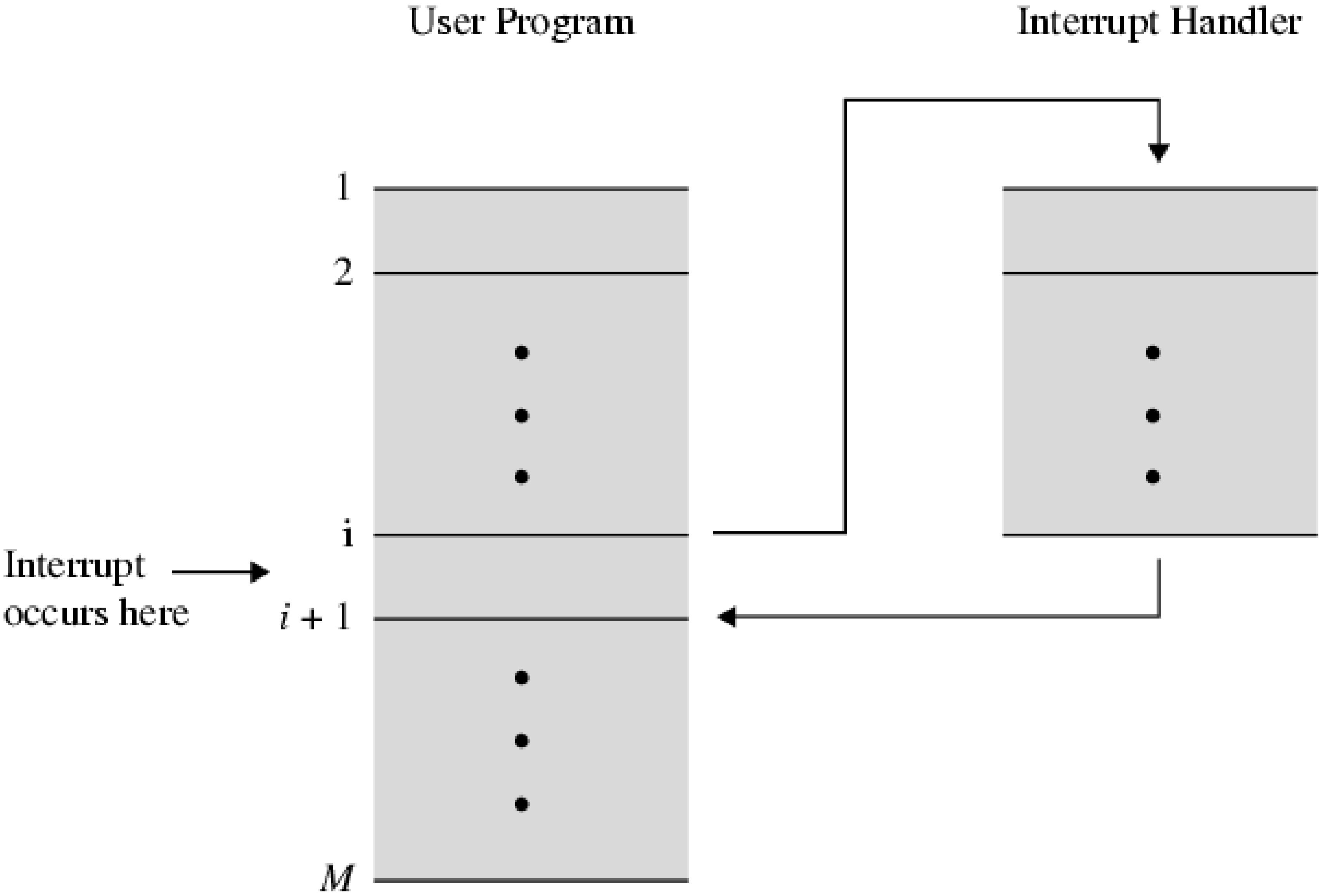
Interrupts – Changing Program Flow

- Mechanism by which other modules (e.g. I/O) may interrupt normal sequence of processing
- Program
  - e.g. overflow, division by zero
- Timer
  - Generated by internal processor timer
  - Used in pre-emptive multi-tasking
- I/O
  - from I/O controller
- Hardware failure
  - e.g. memory parity error

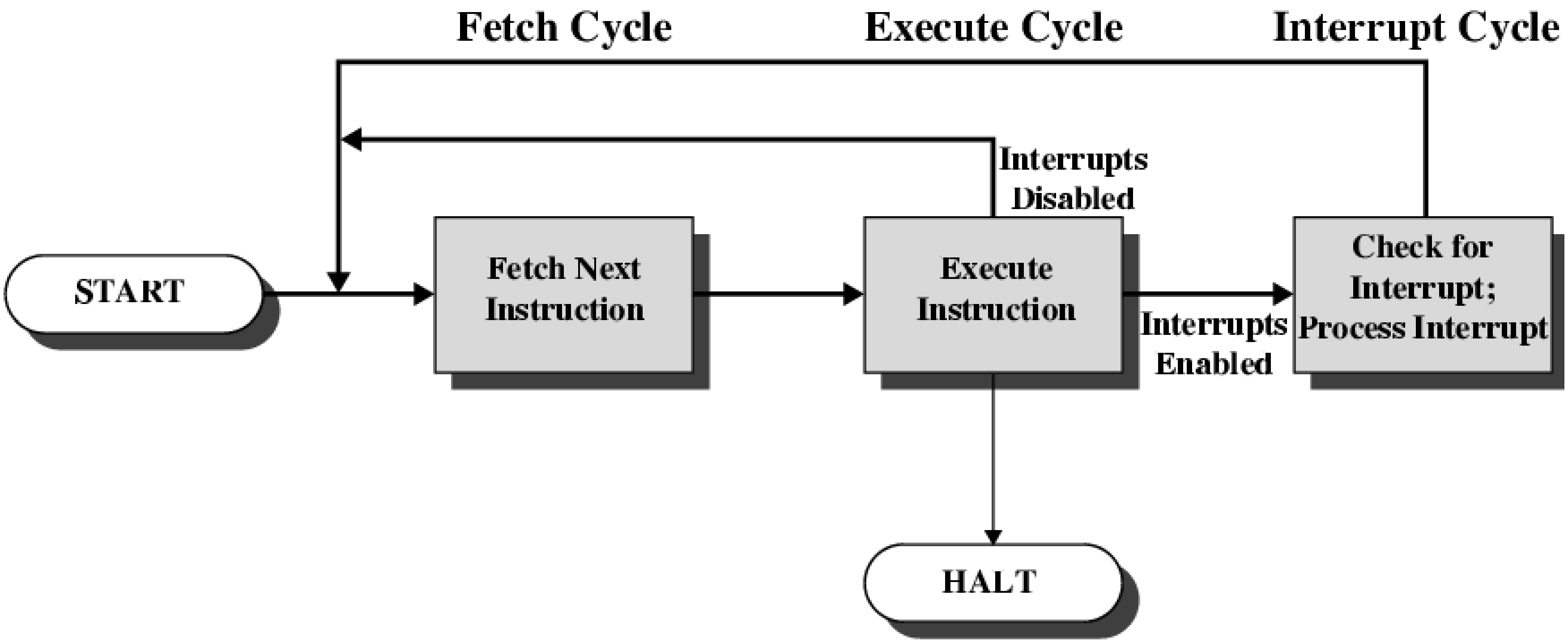
Interrupt Cycle

- Added to instruction cycle
- Processor checks for interrupt
  - Indicated by an interrupt signal
- If no interrupt, fetch next instruction
- If interrupt pending:
  - Suspend execution of current program
  - Save context
  - Set PC to start address of interrupt handler routine
  - Process interrupt
  - Restore context and continue interrupted program

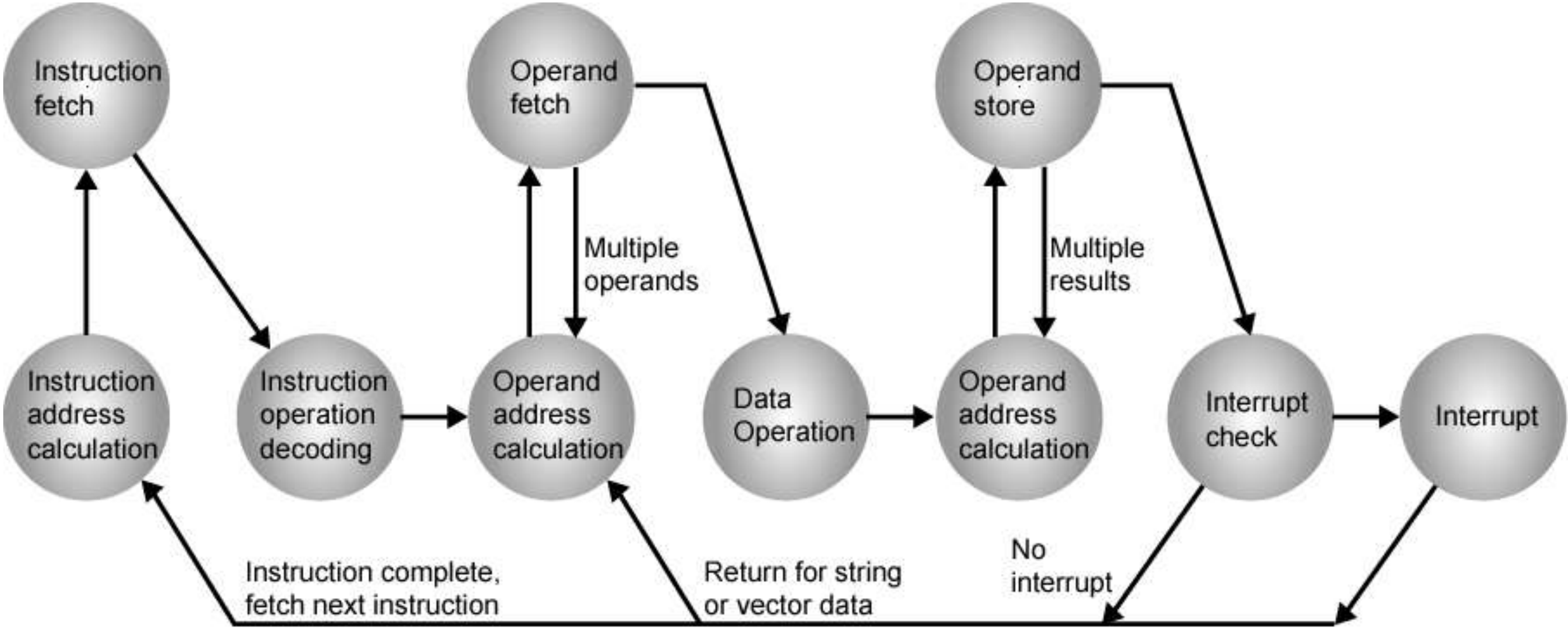
Transfer of Control via Interrupts



Instruction Cycle with Interrupts



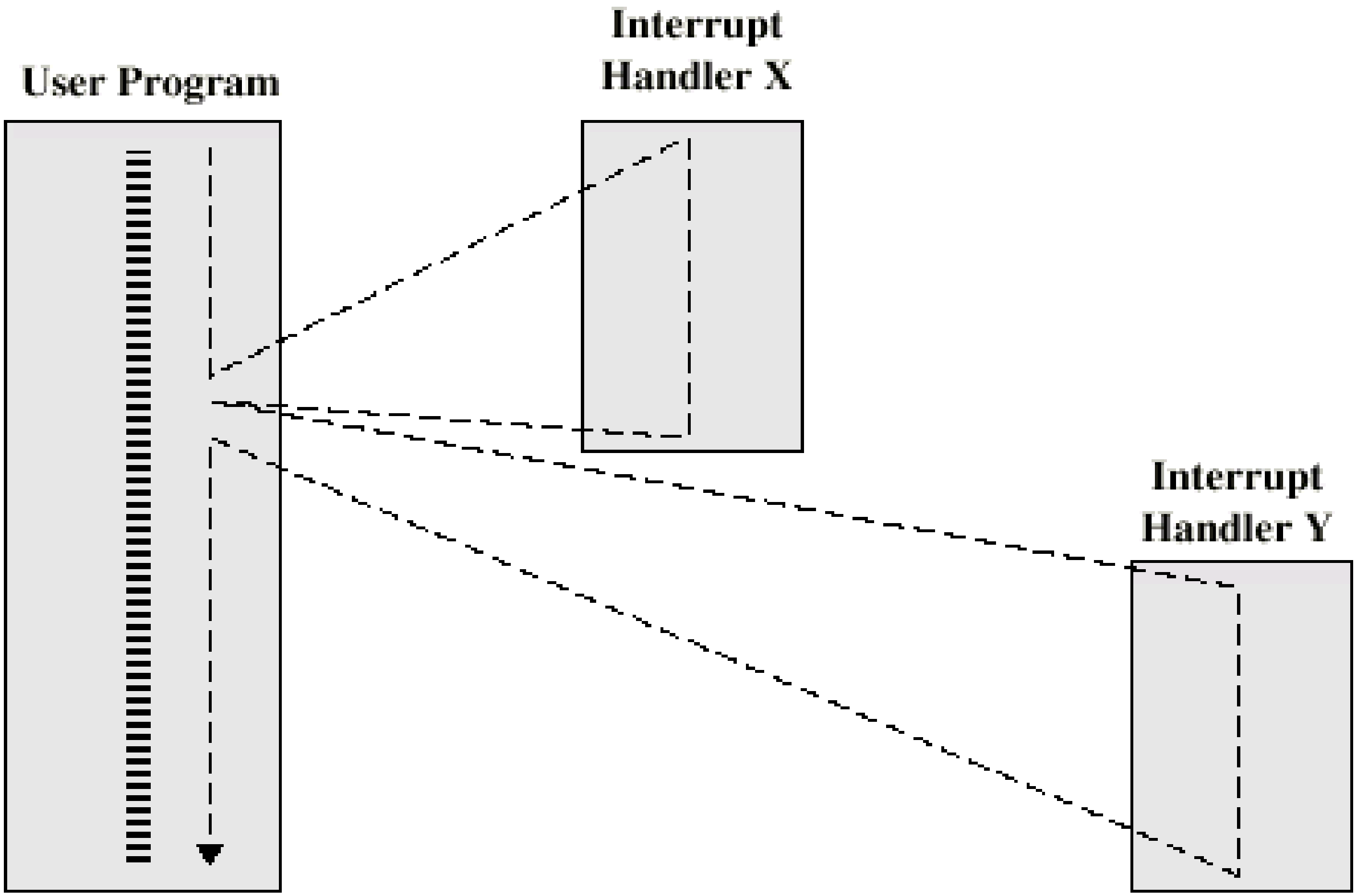
Instruction Cycle (with Interrupts) - State Diagram


Instruction fetch
Operand fetch
Operand store
Multiple operands
Multiple results
Instruction address calculation
Instruction operation decoding
Operand address calculation
Data Operation
Operand address calculation
Interrupt check
Interrupt
Instruction complete, fetch next instruction
Return for string or vector data
No interrupt

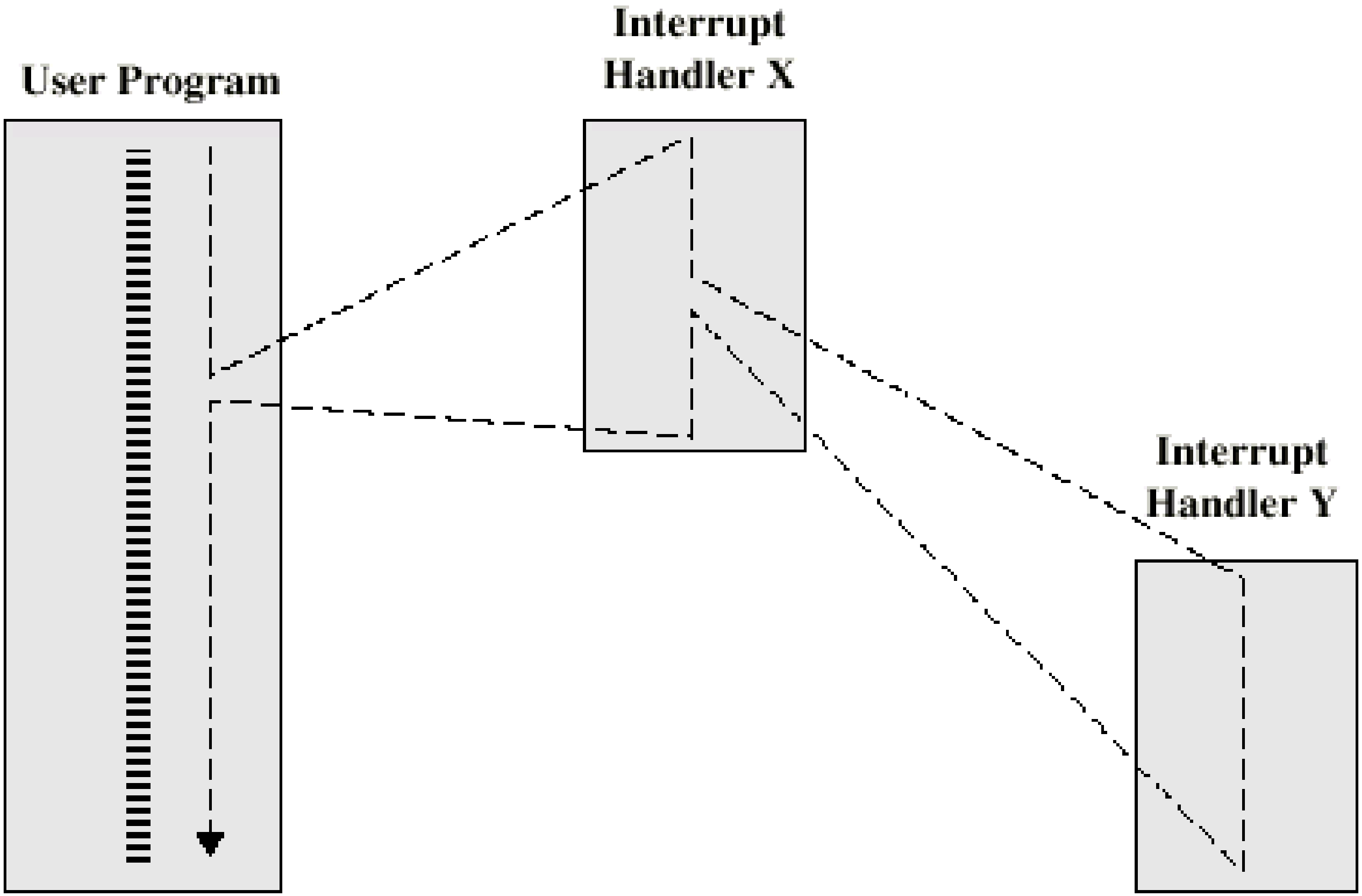
Multiple Interrupts

- Disable interrupts
  - Processor will ignore further interrupts whilst processing one interrupt
  - Interrupts remain pending and are checked after first interrupt has been processed
  - Interrupts handled in sequence as they occur
- Define priorities
  - Low priority interrupts can be interrupted by higher priority interrupts
  - When higher priority interrupt has been processed, processor returns to previous interrupt

Multiple Interrupts - Sequential



Multiple Interrupts – Nested



Input and Output Instructions

I/O Ports

- I/O Devices are connected to the computer through I/O circuits.
- Each circuit contains several registers: **I/O Ports**
- Some ports used for data while others are used for commands.
- Transfer Points between CPU and I/O device.
- Each I/O port:
  - has an address "**I/O Address**"
  - Is connected to the bus system
- I/O Address can only be used with Input / Output instructions.

I/O Port Addresses

- The 8086/8088 supports 64 KB (16 bit) of I/O Port
- Usage vary among computer models
- Some Common I/O Ports:

Port Address	Description
20h-21h	Interrupt Controller
60h-63h	Keyboard Controller
320h-32Fh	Hard Disk

I/O Instructions

- CPU communicates with the peripherals through I/O registers called **I/O Ports**.
- Two instructions to access ports directly.
  - IN
  - OUT
- But most application programs do not use IN and OUT:
  - Port addresses vary among computer models
  - Easier to program by using services routines
- Categories of I/O Service Routines
  - BIOS
    - Stored in ROM and interact directly with I/O ports.
  - DOS
  - More complex tasks like printing a character string

The INT (Interrupt) instruction

- Syntax:

  INT interrupt_number

  - Where interrupt_number specifies a routine.
- Examples

  INT 16h

  - Invokes a BIOS routine that performs keyboard input.

  INT 21h

  - Invoke DOS functions depending on function number present in AH register.

Function No.	Routine
1	Single-key input
2	Single-character output
9	Character string output

Single-Key Input

- AH = 1
- AL = ASCII code if character key is pressed

  = 0 if non-character key is pressed

```
MOV AH,1
INT 21h
```

Single-character output

- AH = 2
- DL = ASCII code of the display character or control character
- AL = ASCII code of the display character or control character

```
MOV AH,2
MOV DL,'?'
INT 21h
```

Control Characters

ASCII Code (Hex)	Symbol	Function
7	BEL	Beep (sound a to e)
8	BS	Backspace
9	HT	Tab
A	LF	Line feed (new line)
D	CR	Carriage return (start of current line)

Sample Programs

Input & Output

- In 8086 assembly language, we use a software interrupt mechanism for I/O.
- An interrupt signals the processor to suspend its current activity (i.e. your running program) and to pass control to an interrupt service program (i.e. part of the operating system).
- A software interrupt is one generated by a program (as opposed to one generated by hardware).
- The 8086 **INT** instruction generates a software interrupt.
- For I/O and some other operations, the number used is **21h**.

Character Input

To read a character from the keyboard:

```
MOV AH, 1
INT 21h
; character is stored in AL
```

Character Output

To display the character ’a’ on the screen:

```
MOV DL, ‘a‘
MOV AH, 2
INT 21h
```

Reading and displaying a character:

```
MOV AH, 1
INT 21h
MOV DL, AL
MOV AH, 2
INT 21h
```

Program 1: Hello World!

```
title Hello World Program          (hello.asm)
; This program displays “Hello, world!”
.model small
.stack 100h
.data
message db “Hello, world!”,0dh,0ah,’$’
.code
main proc
    mov  ax,@data
    mov  ds,ax
    mov  ah,9
    mov  dx,offset message
    int  21h

    mov  ax,4C00h
    int  21h
main endp
end main
```

program title (comment)

```
title Hello World Program            (hello.asm)

; This program displays “Hello, world!”

.model small

.stack 100h
```

comment line

memory model

set the stack size

```
.data
message db “Hello, world!”,0dh,0ah,’$’

.code
main proc
    mov  ax,@data
    mov  ds,ax
    mov  ah,9
    mov  dx,offset message
    int  21h

    mov  ax,4C00h
    int  21h
main endp
end main
```

starts the data segment

starts the code segment

sets DS to the offset of the data segment

calls DOS display function 9

halts program

Program 2: Echo

```
TITLE MY First Program
.MODEL SMALL
.STACK 100H
.CODE
;display prompt
    MOV AH, 2     ;display character function
    MOV DL, '?'  ;character is '?'
    INT 21H       ;display it
;input a character
    MOV AH, 1     ;read character function
    INT 21H       ;character in AL
    MOV BL,AL     ;save it in BL
```

Contd..

```
;go to a new line
    MOV AH, 2
    MOV DL, 0DH
    INT 21H
    MOV DL, 0AH
    INT 21H
;display character
    MOV DL, BL
    INT 21H
;return to DOS
    MOV AH, 4CH
    INT 21H
```

Program 3: Add

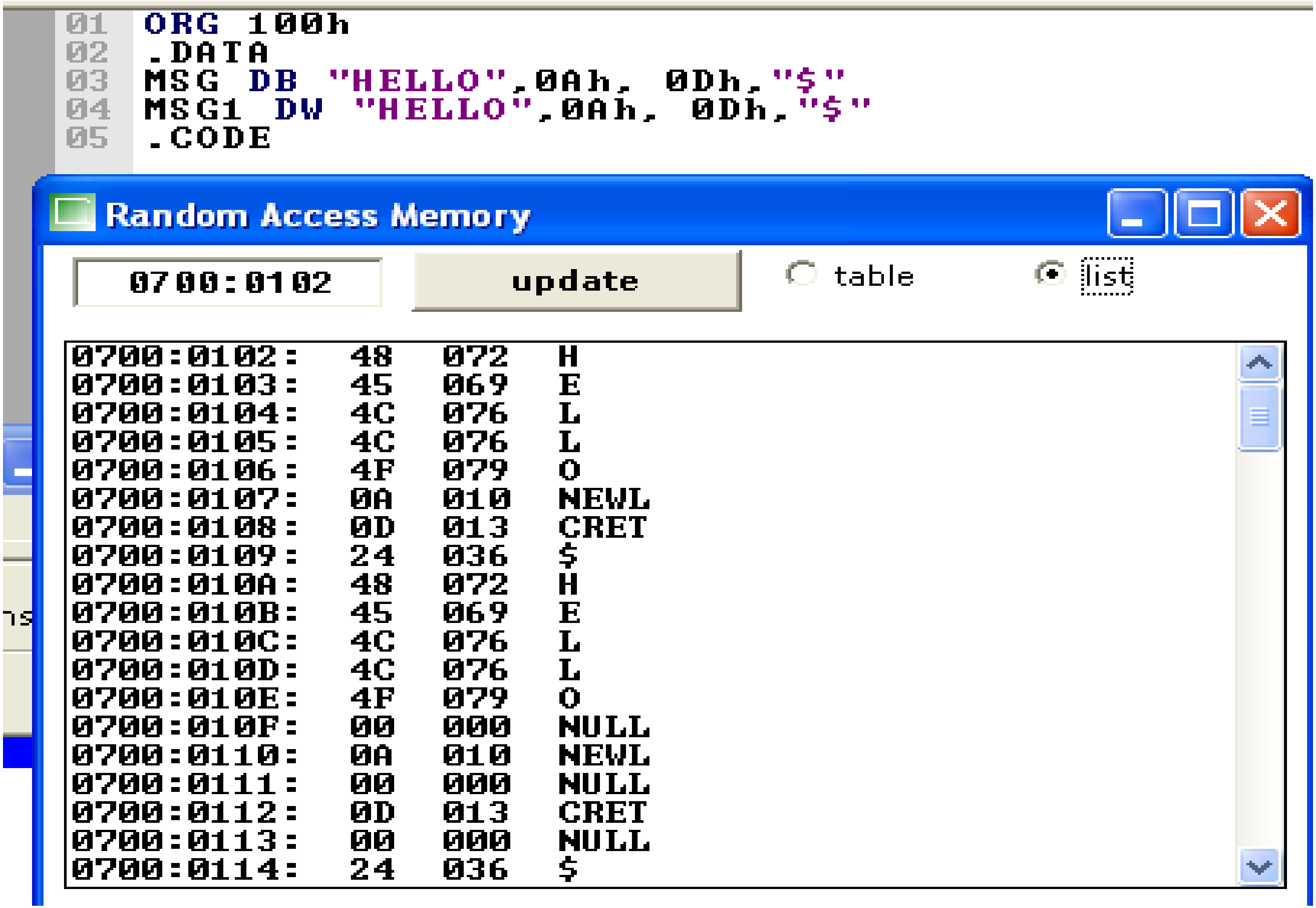
```
.DATA
A DW 2
B DW 5
SUM DW ?
.CODE
;add the numbers
    MOV AX, A
    ADD AX, B
    MOV SUM, AX
;exit to DOS
    MOV AX, 4C00H
    INT 21H
```

Program 4: Lower To Upper case

```
TITLE Case Conversion Program
.MODEL SMALL
.STACK 100H
.DATA
     CR EQU 0DH
     LF EQU 0AH
MSG1 DB 'Enter a Lowe Case Letter: $'
MSG2 DB 0DH, 0AH, 'In Upper Case It is: '
CHAR DB ?,'$‘
.CODE
;initialize DS
    MOV AX, @DATA    ;get data segment
    MOV DS, AX       ;initialize DS
```

Contd..

```
;print user prompt
    LEA DX, MSG1     ;get first message
    MOV AH, 9        ;display string function
    INT 21H          ;display first message
;input a character and convert to upper case
    MOV AH, 1        ;read character function
    INT 21H          ;read snall letter into AL
    SUB AL, 20H      ;convert it into uppercase
    MOV CHAR, AL     ;and store it
;display on the next line
    LEA DX, MSG2     ;get second message
    MOV AH, 9        ;display string function
    INT 21H          ;display message and upper case letter in front
;
```



```
ORG 100h
.DATA
MSG DB "HELLO",0Ah, 0Dh,"$"
MSG1 DW "HELLO",0Ah, 0Dh,"$"
.CODE
```

0700:0102

○ table ● list

```
0700:0102:   48   072   H
0700:0103:   45   069   E
0700:0104:   4C   076   L
0700:0105:   4C   076   L
0700:0106:   4F   079   O
0700:0107:   0A   010   NEWL
0700:0108:   0D   013   CRET
0700:0109:   24   036   $
0700:010A:   48   072   H
0700:010B:   45   069   E
0700:010C:   4C   076   L
0700:010D:   4C   076   L
0700:010E:   4F   079   O
0700:010F:   00   000   NULL
0700:0110:   0A   010   NEWL
0700:0111:   00   000   NULL
0700:0112:   0D   013   CRET
0700:0113:   00   000   NULL
0700:0114:   24   036   $
```