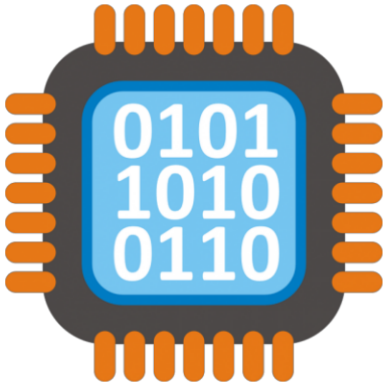
Secure Assembly Coding

Week # 9 Lectures



Dr. Qasem Abu Al-Haija,
Department of Cybersecurity,

Input / Output Functions

Input / Output Functions Using DOS Interrupt 21H

SERVICE	DESCRIPTION	EXAMPLE
1	Read one character from the keyboard	MOV AH, 1 INT 21H AL= ASCI CHARCTER

2	Display a character	MOV AH, 2
		MOV DL, 35h
		Int 21H

9	Display a String	STR DB “Hello$” MOV AH, 9 MOV DX,OFFSET STR INT 21H
0AH	Reading String	inputarea db 10,0,10 dup(' ') mov dx,offset inputarea mov ah,0A int 21h

At the beginning, lets have a look at ASCII table

Character	Decimal	Hex	Binary	Octal
0	48	30	110000	60
1	49	31	110001	61
2	50	32	110010	62
3	51	33	110011	63
4	52	34	110100	64
5	53	35	110101	65
6	54	36	110110	66
7	55	37	110111	67
8	56	38	111000	70
9	57	39	111001	71

Character	Decimal	Hex	Binary	Octal
A	65	41	1000001	101
B	66	42	1000010	102
C	67	43	1000011	103
D	68	44	1000100	104
E	69	45	1000101	105
F	70	46	1000110	106
G	71	47	1000111	107
H	72	48	1001000	110
I	73	49	1001001	111
J	74	4A	1001010	112
K	75	4B	1001011	113
L	76	4C	1001100	114
M	77	4D	1001101	115
N	78	4E	1001110	116
O	79	4F	1001111	117
P	80	50	1010000	120
Q	81	51	1010001	121
R	82	52	1010010	122
S	83	53	1010011	123
T	84	54	1010100	124
U	85	55	1010101	125
V	86	56	1010110	126
W	87	57	1010111	127
X	88	58	1011000	130
Y	89	59	1011001	131
Z	90	5A	1011010	132

Character	Decimal	Hex	Binary	Octal
a	97	61	1100001	141
b	98	62	1100010	142
c	99	63	1100011	143
d	100	64	1100100	144
e	101	65	1100101	145
f	102	66	1100110	146
g	103	67	1100111	147
h	104	68	1101000	150
i	105	69	1101001	151
j	106	6A	1101010	152
k	107	6B	1101011	153
l	108	6C	1101100	154
m	109	6D	1101101	155
n	110	6E	1101110	156
o	111	6F	1101111	157
p	112	70	1110000	160
q	113	71	1110001	161
r	114	72	1110010	162
s	115	73	1110011	163
t	116	74	1110100	164
u	117	75	1110101	165
v	118	76	1110110	166
w	119	77	1110111	167
x	120	78	1111000	170
y	121	79	1111001	171
z	122	7A	1111010	172

Character	Decimal	Hex	Binary	Octal
BEL - bell	7	7	111	7
BS - backspace	8	8	1000	10
TAB - horizontal tab	9	9	1001	11
LF - line feed	10	0A	1010	12
VT - vertical tab	11	0B	1011	13
FF - form feed	12	0C	1100	14
CR - carriage return	13	0D	1101	15

Input / Output devices deal with ASCII code only

- **When you read a character from the keyboard… The Ascii code of the character is read… for example if the user presses on the A character, the code 41H is entered not the letter A. If the user presses on the character 5 from the keyboard, the code 35H is entered, and so on.**

- **If you are writing a program to display a character on the screen, then you have to send the ascii code of the character…. For example to display the letter B, you send the code 42H…If you want to display the character 6 on the screen, you send the code 36H, and so on…**

- **If you examine the ASCII code table you will find that there is a difference of 20H between Upper case letters and smaller case letters. For example "A" = 41H while "a" = 61H, "B" = 42H while "b" = 62….**
- **You can use the difference 20H in your programs to convert lower case letters to upper case, and vice versa (add 20H to convert upper to lower, and subtract 20H to convert from lower to upper).**

- **If you also look at the Ascii code table and look at the Integers from 0 – 9, you will see that they start with 30H for zero and end with 39H for the character 9….. So when you read a character that represent a digit in your program, you must subtract 30H from it to treat it as a digit not as a character so you can include it in your calculation.**
- **If you want to print a number in your code, say the result of adding two numbers (3+5 = 8), to print the result (8) on the screen you should convert it to ascii code by adding 30H to it.**

Defining messages in the data segment first:

- All messages are strings, so they must be defined using DB
- All messages that are to be printed must end with a "$" to indicate the end of the string
- You can add the ASCII codes 0AH, and 0DH at the beginning or at the end of the message to print a new line

```
.Data
    Message1 db "Hello and welcome to my program", 0Ah, 0DH, "$"
    Message2 db 0Ah, 0Dh, "Please enter a number:$"
```

- You can define a message with no text, but with 0AH and 0DH to print a blank line when you need it

```
    Enter1 db 0AH, 0DH,"$"
```

EXAMPLE

Write a program which prints a welcome message to the user, then asks the user to enter two numbers, then add the two numbers and print results on the screen. Make sure the total is less than 10 (Why?).

First thing to do is to prepare the required messages in the data segment

```
.data
Welcome db “hello and welcome to my program”, 0aH,0dh,”$”
Enter1 db 0ah,0dh,”$”
In1 db “please enter a number:$”
In2 db “ please enter a second number:$”
Results db “The total is:$”
Num1 db ?
Num2 db ?
Total db 0
```

Second thing to do is to write the required assembly instructions in the code segment

```
.Code
Mov ah, 9
Mov dx, offset welcome    ; display welcome message
Int 21h

Mov ah,9
Mov dx, offset in1        ; Display message to user to Enter  a number (single digit)
Int 21h

Mov ah, 1                 ; read the number as ASCII code
Int 21h

Sub al, 30h               ; convert number from ASCII to a real digit, store it in num1
Mov num1, al

Mov ah,9                  ; Display a blank line
Mov dx, offset enter1
Int 21h
```

Continue on next page

```
Mov ah,9
Mov dx, offset in2 ; display a message to user to enter second number
Int 21h

Mov ah, 1                ; read second number, and convert it to a real digit
Int 21h
Sub al, 30h
Mov num2, al             ; store the digit in variable num2

Mov ah,9                 ; display a blank line
Mov dx, offset enter1
Int 21h

 Mov al, num2
Add al, num1             ; note that the second number is still in AL, so Add num1 to al
Mov total, al            ; and store results in al, then store results in Total

Add total, 30h           ; prepare total for printing – convert it to Ascii
Mov ah,9                 ; print the message result
Mov dx, offset results
Int 21h

Mov ah, 2                ; print the total on the screen
Mov dl, total
Int 21h
```

New Line Function

```
call NewL


NewL:                                   ; New Line funtion
  Mov ah,9                              ; Display a blank line
  Mov dx, offset enter1
  Int 21h
ret
```

Clear Screen Function

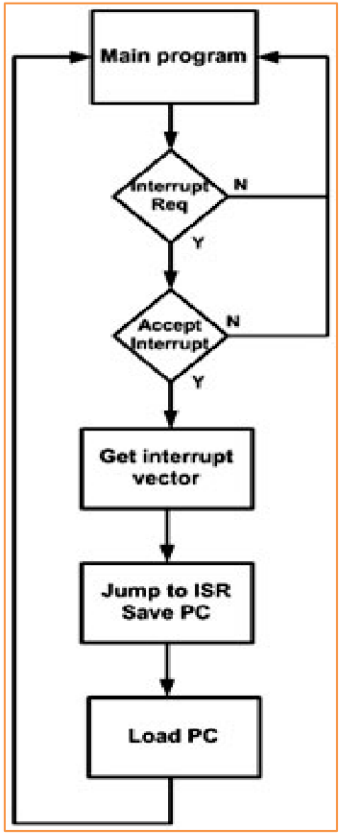
```
call cls

cls:
  mov ah, 00h
  mov al,  03h          ; text mode 80x25 16 colors
  int 10h
ret
```

Interrupt Instructions

8086 Instruction Set

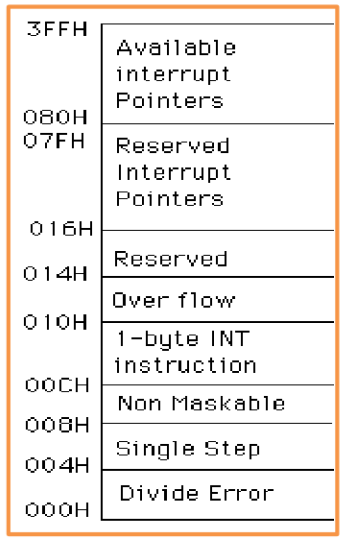
Group 9: Interrupt Instructions

- In addition to the HW interrupts (NMI/INTR), 8086 allows for SW interrupts by execution of instruction INT N.
  - N is the software interrupt that should be generated (0-255).
- Execution of INT n causes 8086 to:
  - Push current CS, IP and Flags onto Stack.
  - Loads CS and IP with new values based on interrupt type N *(Using interrupt vector table IVT).*
  - An interrupt service routine is written at this new address.
  - IRET at the end of ISR transfers control to main program: *(by popping old CS, IP, and flags from the stack).*
- Example: Overflow (type 4)
  - Occurs if OF is set and INTO inst is executed.
  - INTO: Useful after exe of a signed arithmetic (as IMUL).



8086 Interrupt Instructions

INT n (n can be 0-255_{10})	Software interrupt instructions (INT 32_{10} – 255_{10} available to the user.)
INTO	Interrupt on overflow
IRET	Interrupt return



Interrupt Vector Table

8086 Instruction Set

Group 9 : Interrupt Instructions

Example:

```
; add the numbers
      MOV AX, A
       ADD AX, B
       MOV SUM, AX
; exit to DOS
      MOV AX, 4C00H
      INT 21H
MAIN ENDP
      END MAIN
```

Example:

STI / CLI

Two instructions used to enable/disable the maskable interrupts (HW interrupts from INTR pin)

Mnemonic	Meaning	Format	Operation	Flags Affected
CLI	Clear interrupt flag	CLI	0 → (IF)	IF
STI	Set interrupt flag	STI	1 → (IF)	IF

Example:

INT 03H

Predefined interrupt to set a breakpoint in the program (pausing place in a program, put in place for debugging purposes.)

We can also define our own interrupts or use BIOS interrupts

More about interrupts on Chapter 5

Main Sources for these slides

- *K. R. Irvine. Assembly Language for x86 Processors, 8th edition, Prentice-Hall (Pearson Education), June 2019. ISBN: 978-0135381656.*

- *B. Dang, A. Gazet, E. Bachaalany. Practical Reverse Engineering: x86, x64, ARM, Windows® Kernel, Reversing Tools, and Obfuscation. John Wiley & Sons, June 2014. ISBN: 978-1-118-78731-1*

- *Qasem Abu Al-Haija, "Microprocessor Systems", King Faisal University, Saudi Arabia*

- *Ghassan Issa, "Computer Organization", Petra University, Jordan.*

Thank you