A86 Examples

CMPE 230 - Spring 2022

Gökçe Uludoğan

Based on the slides by Abdullatif Köksal, with his permission.

A86 Assembler

- **Variable definition**

```
READ_FLAG     DB   0        ; Byte variable, initial value 0
MSG           DB   "Hello"  ; We can define string variable with a value
```

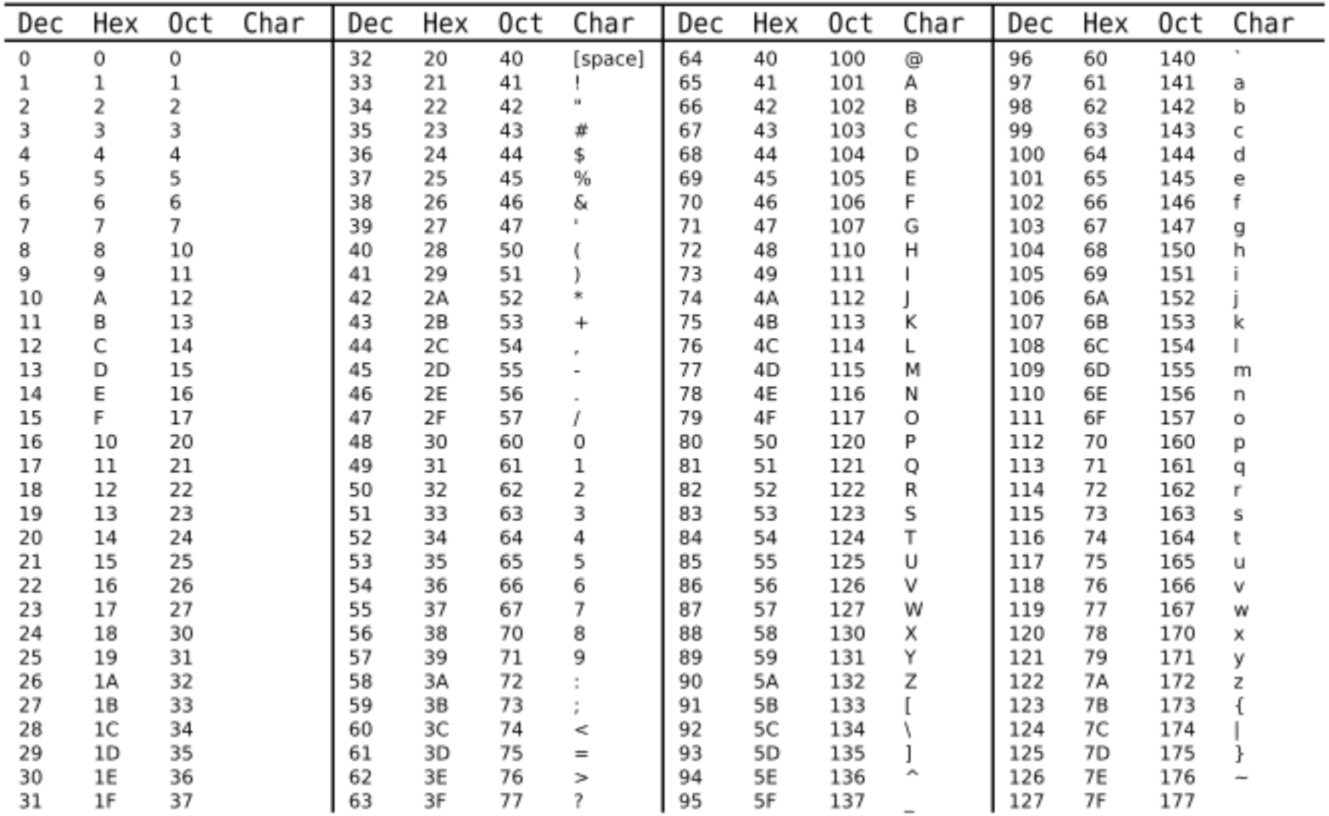
- **MOV**

```
MOV BH,DL  ; This copies the bottom byte of the DX register into the top byte of BX.
MOV AH,12  ; This puts the value 12 decimal into the top half of the AX register.
```

- **ASCII Table**

  You can use ASCII table to find character encodings.

ASCII Table



Dec	Hex	Oct	Char	Dec	Hex	Oct	Char	Dec	Hex	Oct	Char	Dec	Hex	Oct	Char
0	0	0		32	20	40	[space]	64	40	100	@	96	60	140	`
1	1	1		33	21	41	!	65	41	101	A	97	61	141	a
2	2	2		34	22	42	"	66	42	102	B	98	62	142	b
3	3	3		35	23	43	#	67	43	103	C	99	63	143	c
4	4	4		36	24	44	$	68	44	104	D	100	64	144	d
5	5	5		37	25	45	%	69	45	105	E	101	65	145	e
6	6	6		38	26	46	&	70	46	106	F	102	66	146	f
7	7	7		39	27	47	'	71	47	107	G	103	67	147	g
8	8	10		40	28	50	(	72	48	110	H	104	68	150	h
9	9	11		41	29	51	)	73	49	111	I	105	69	151	i
10	A	12		42	2A	52	*	74	4A	112	J	106	6A	152	j
11	B	13		43	2B	53	+	75	4B	113	K	107	6B	153	k
12	C	14		44	2C	54	,	76	4C	114	L	108	6C	154	l
13	D	15		45	2D	55	-	77	4D	115	M	109	6D	155	m
14	E	16		46	2E	56	.	78	4E	116	N	110	6E	156	n
15	F	17		47	2F	57	/	79	4F	117	O	111	6F	157	o
16	10	20		48	30	60	0	80	50	120	P	112	70	160	p
17	11	21		49	31	61	1	81	51	121	Q	113	71	161	q
18	12	22		50	32	62	2	82	52	122	R	114	72	162	r
19	13	23		51	33	63	3	83	53	123	S	115	73	163	s
20	14	24		52	34	64	4	84	54	124	T	116	74	164	t
21	15	25		53	35	65	5	85	55	125	U	117	75	165	u
22	16	26		54	36	66	6	86	56	126	V	118	76	166	v
23	17	27		55	37	67	7	87	57	127	W	119	77	167	w
24	18	30		56	38	70	8	88	58	130	X	120	78	170	x
25	19	31		57	39	71	9	89	59	131	Y	121	79	171	y
26	1A	32		58	3A	72	:	90	5A	132	Z	122	7A	172	z
27	1B	33		59	3B	73	;	91	5B	133	[	123	7B	173	{
28	1C	34		60	3C	74	<	92	5C	134	\	124	7C	174	|
29	1D	35		61	3D	75	=	93	5D	135	]	125	7D	175	}
30	1E	36		62	3E	76	>	94	5E	136	^	126	7E	176	~
31	1F	37		63	3F	77	?	95	5F	137	_	127	7F	177	

A86 Assembler

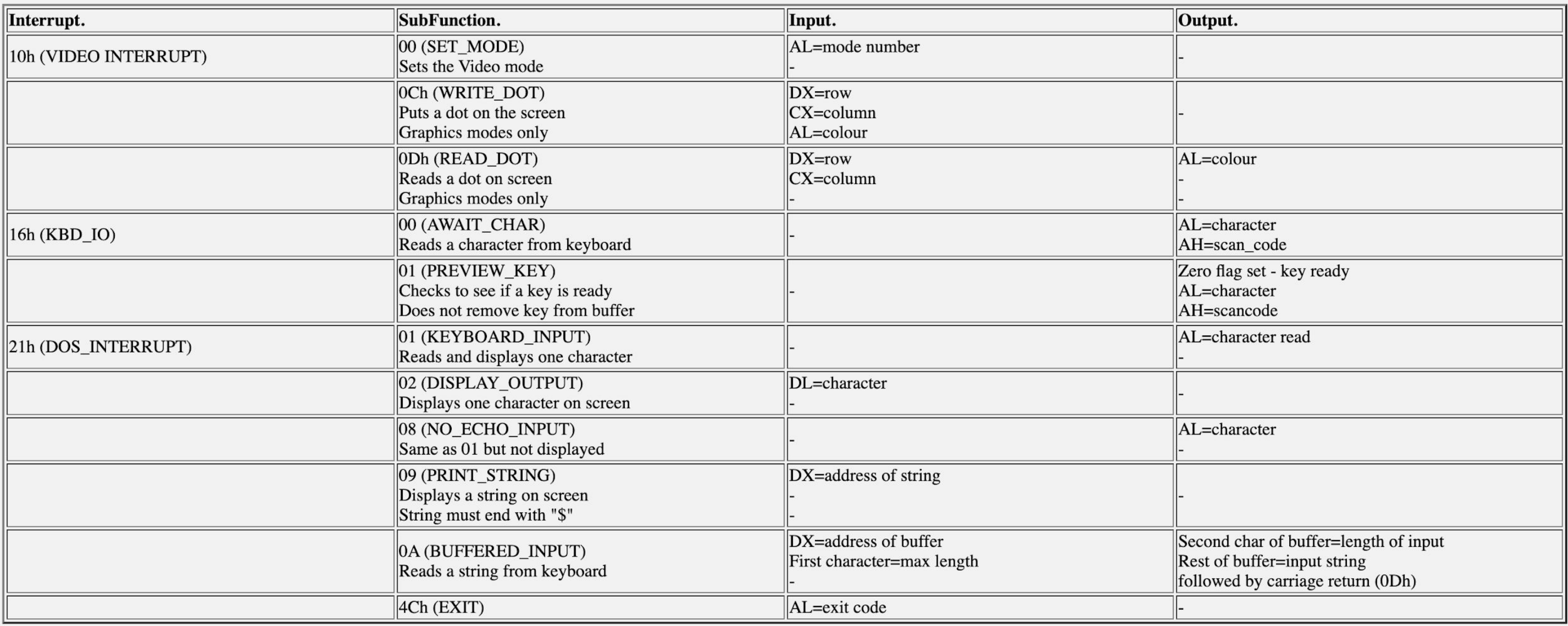
- **STACK**

```
PUSH AX ; AX onto stack
POP BX  ; pop into BX
```

- **INT**

  We use interrupt function to do several things like printing character/string, reading character, exit etc.

INT: Interrupt - Useful Subfunctions



Interrupt.	SubFunction.	Input.	Output.
10h (VIDEO INTERRUPT)	00 (SET_MODE) Sets the Video mode	AL=mode number -	-

	0Ch (WRITE_DOT)
	Puts a dot on the screen
	Graphics modes only
		DX=row
		CX=column
		AL=colour
			-

	0Dh (READ_DOT)
	Reads a dot on screen
	Graphics modes only
		DX=row
		CX=column
		-
			AL=colour
			-
			-

16h (KBD_IO)	00 (AWAIT_CHAR)
	Reads a character from keyboard
		-	AL=character
			AH=scan_code

	01 (PREVIEW_KEY)
	Checks to see if a key is ready
	Does not remove key from buffer
		-	Zero flag set - key ready
			AL=character
			AH=scancode

21h (DOS_INTERRUPT)	01 (KEYBOARD_INPUT) Reads and displays one character	-	AL=character read -
	02 (DISPLAY_OUTPUT) Displays one character on screen	DL=character -	-
	08 (NO_ECHO_INPUT) Same as 01 but not displayed	-	AL=character -
	09 (PRINT_STRING) Displays a string on screen String must end with "$"	DX=address of string - -	-
	0A (BUFFERED_INPUT) Reads a string from keyboard	DX=address of buffer First character=max length -	Second char of buffer=length of input Rest of buffer=input string followed by carriage return (0Dh)
	4Ch (EXIT)	AL=exit code	-

http://www.csn.ul.ie/~darkstar/assembler/intlist.html

A86 Assembler

- **Labels**

  Labels are indicators of upcoming code segment.

- **CMP**

  We use cmp to compare different values. Then, we can use outcome of the comparison via jump function.(equal, not equal, greater than, less than)

- **JMP**

  We use JMP to directly jump.

  ```
  JMP - Will jump no matter what, doesn't check conditions.
  ```

A86 Assembler

- **Conditional Jumps**

```
JE  - Will jump if compared things are equal.

JNE - Will jump if comparison is not equal.

JA/JG  - Will jump if the first thing is greater.

JB/JL  - Will jump if the first thing is less.
```

For difference between JA and JG: https://stackoverflow.com/questions/20906639/difference-between-ja-and-jg-in-assembly

A86 Assembler

- **Arithmetic Operations**

```
INC AL       ; Increments value in AL by one and writes in AL again.

DEC AL       ; Decrements value in AL by one and writes in AL again.

ADD AX, BX   ; Add value in AX and value in BX and writes into AX

SUB AX, BX   ; Value in AX minus value in BX and writes into AX. AX = AX - BX

MUL CH       ;(AL*CH -> AX)  Multiplication uses AL by default as
multiplicand and writes into AX for 8-bit values:

DIV 7        ; AH = AX % 7, AL = AX / 7 Division uses AX by default as
dividend and writes result into AL and remainder in AH for 8-bit values:
```

MUL and DIV use different register when you use different values than 8-bit values. Be careful about that.

A86 Assembler

- Arithmetic Operations

```
AND AX, BX    ; bitwise and
OR  AX, BX    ; bitwise or
XOR AX, BX    ; bitwise xor
NOT AX        ; bitwise not
```

Manual and References

If you have questions regarding functions and A86 assembler, you can check out Manual and references for different tutorials.

Manual:

https://fruttenboel.verhoeven272.nl/asm/a86man.html

Tutorials:

http://www.csn.ul.ie/~darkstar/assembler

https://patater.com/gbaguy/x86asm.htm

References

1. http://www.csn.ul.ie/~darkstar/assembler
2. https://fruttenboel.verhoeven272.nl/asm/a86man.htm
3. https://www.wikiwand.com/en/Assembly_language
4. https://www.wikiwand.com/en/Machine_code
5. https://cysecguide.blogspot.com/2016/12/ascii-code-table.html
6. https://patater.com/gbaguy/x86asm.htm