INTRODUCTION ABOUT ASSEMBLY

Grade

Participation	5
Lab attendance and lab work	5
Homework	5
Final lab	15

What is assembly language

- Assembly language is a low-level programming language for a computer.
- Each statement is [almost] exactly equivalent to one machine instruction.
- Assembly language is converted into executable machine code by using assembler like NASM, MASM, etc.

Advantages of Assembly Language

- Studying assembly language is it possible to gain a feeling for the way the computer "thinks" and why certain things happen the way they do inside the computer.
- Writing assembly language programs is efficiency because assembly language is so close to machine language so produces a faster , shorter machine language program.
- It allows hardware-specific complex jobs in an easier way.
- It is suitable for time-critical jobs.

Difference between high level language and assembly level language

	High level language	Assembly language
	Easier to read and understand	Difficult to read and understand
Portable	Yes	No , designed for a specific family of processors
Number of equivalent machine code	Each statement need many machine instruction	Each statement is [almost] one machine instruction

Example	X=6+Y	MOV AX,6
		ADD AX,Y
		MOV A,AX

Reference:

- Assembly Language Programming and Organization of the IBM PC.
- Tutorialspoint website “https://www.tutorialspoint.com/”.

Requirements for coding in Assembly Language

Outline

- Assembly Language Statements
- Defining Types of data

➢Assembly Language Statements

• Programs consist of statements, one per line.
• Statements is either **instruction** or **assembler directive**.
• Instruction , which the assembler translate into machine code.
• Assembler directive, which instructs the assembler to perform some specific task.
• Both instructions and directives have up to four fields:

[identifier] **operation** **[operand(s)]** **[comment]**

• At least one blank or tab character must separate the fields.
• The fields do not have to be aligned in a particular column, but they must appear in the above order.

➢Assembly Language Statements

- An example of an instruction:

  START: MOV CX,5 ; initialize counter

- An example of an assembler directive:

  MAIN PROC

Identifier field

- The identifier is used for instruction labels, procedure names and variable name.
- Can be from 1 to 31 characters long (not case sensitive).
- May consist of letters, digits, and the special characters ? . @ _ $ % (Thus, embedded blanks are not allowed).
- Names may not begin with a digit.
- If a dot is used, it must be the first character.

The assembler does not differentiate between uppercase and lowercase.

Identifiers

- Examples:
  - COUNTER1 ✓
  - 2abc ✗ Begins with a digit
  - @CHARACTER ✓
  - A45. 28 ✗ . Not first character
  - TWO WORDS ✗ Contains a blank
  - STD_NUM ✓
  - .TEST ✓
  - YOU&ME ✗ Contains an illegal character

Operation field

- **For an instruction**, the operation field contains a symbolic operation code (opcode). The assembler translates a symbolic opcode into machine language. For example: mov, add , sub.

- **For an assembler directive**, the operation field contains a pseudo-operation code (pseudo-op). pseudo-op are not translated into machine code, they simply tell assembler to do something. For example: DB , DW , PROC.

Operand field

- **For an instruction**, the operand fields specifies the data that are to be acted on by the operation. An instruction may have zero, one, two operands.

- **For an assembler directive**, the operand fields usually contains more information about the directive.

Comment field

• The comment field of a statement is used by the programmer to say something about what the statement does.

• A semicolon marks the beginning of this field, and the assembler ignores anything typed after the semicolon.

• It is almost impossible to understand an assembly language program without comments.

• Good programming practice dictates a comment on almost every line.

• Examples:

•MOV CX, 0 ; CX counts terms, initially 0

➢ Defining Types of data

[name]	Dn	expression

Name: a program that references a data item does so by means of name

Dn (Directive): define the data item – see next slide—

Expression: is an operand may specify an uninitialized value or constant value
an uninitialized value defined by item ?
EXAMPLE :

DATAX DB ?

Defining Types of data (Directive):

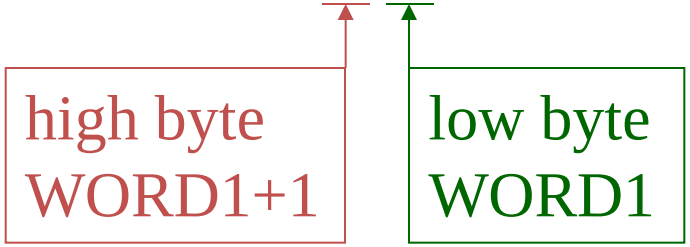
Pseudo-op	Stands for
DB	Define Byte
DW	Define Word
DD	Define Doubleword
DQ	Define Quadword
DT	Define Tenbytes

Defining Types of data -Examples

ALPHA DB 4	a memory byte is associated with the name ALPHA, and initialized to 4
BYT DB ?	A memory byte is associated with the name BYT, and uninitialized.
WRD DW -2	a memory word is associated with the name WRD, and initialized to -2.

High and Low Bytes of a Word

- WORD1 DW 1234H



Defining Types of data

The decimal range (fit a byte):

- Unsigned representation: 0 to 255
- Signed representation: -128 to 127

The decimal range (fit a word):

- Unsigned representation: 0 to 65535
- Signed representation: -32768 to 32767

Defining Types of data – Array byte

- an array is a sequence of memory bytes or words.

- Example:

  B_ARRAY DB 10H,20H,30H

Symbol	Address	Contents
B_ARRAY	200H	10H
B_ARRAY+1	201H	20H
B_ARRAY+2	202H	30H

Defining Types of data – Array word

- Example:
  W_ARRAY DW 1000,40,29887,329

Symbol	Address	Contents
W_ARRAY	0300H	1000D
W_ARRAY+2	0302H	40D
W_ARRAY+4	0304H	29887D
W_ARRAY+6	0306H	329D

Defining Types of data :The DUP Operator

• It is possible to define arrays whose elements share a common initial value by using the DUP (duplicate) operator.

• Syntax:

[name]	Dn	Repeat-count(exp)

• Example:

DELTA DB 212 DUP (?)	creates an array of 212 uninitialized bytes.
GAMMA DW 100 DUP (0)	set up an array of 100 words, with each entry initialized to 0.

Character String

• ASCII codes can be initialized with a string of characters using single quotes like ‘PC’ or double quotes like “PC”.

• Example:

```
LETTERS    DB      'ABC'
=
LETTERS    DB      41H,42H,43H
```

• Inside a string, the assembler differentiates between upper and lowercase.

• It is possible to combine characters and numbers in one definition:

```
Example: MSG       DB      'HELLO',0AH,0DH, '$'
```

Dec	Hx	Oct	Char	Dec	Hx	Oct	Html	Chr	Dec	Hx	Oct	Html	Chr	Dec	Hx	Oct	Html	Chr
0	0	000	NUL (null)	32	20	040		Space	64	40	100	@	@	96	60	140	`	`
1	1	001	SOH (start of heading)	33	21	041	!	!	65	41	101	A	A	97	61	141	a	a
2	2	002	STX (start of text)	34	22	042	"	"	66	42	102	B	B	98	62	142	b	b
3	3	003	ETX (end of text)	35	23	043	#	#	67	43	103	C	C	99	63	143	c	c
4	4	004	EOT (end of transmission)	36	24	044	$	$	68	44	104	D	D	100	64	144	d	d
5	5	005	ENQ (enquiry)	37	25	045	%	%	69	45	105	E	E	101	65	145	e	e
6	6	006	ACK (acknowledge)	38	26	046	&	&	70	46	106	F	F	102	66	146	f	f
7	7	007	BEL (bell)	39	27	047	'	'	71	47	107	G	G	103	67	147	g	g
8	8	010	BS (backspace)	40	28	050	(	(	72	48	110	H	H	104	68	150	h	h
9	9	011	TAB (horizontal tab)	41	29	051	)	)	73	49	111	I	I	105	69	151	i	i
10	A	012	LF (NL line feed, new line)	42	2A	052	*	*	74	4A	112	J	J	106	6A	152	j	j
11	B	013	VT (vertical tab)	43	2B	053	+	+	75	4B	113	K	K	107	6B	153	k	k
12	C	014	FF (NP form feed, new page)	44	2C	054	,	,	76	4C	114	L	L	108	6C	154	l	l
13	D	015	CR (carriage return)	45	2D	055	-	-	77	4D	115	M	M	109	6D	155	m	m
14	E	016	SO (shift out)	46	2E	056	.	.	78	4E	116	N	N	110	6E	156	n	n
15	F	017	SI (shift in)	47	2F	057	/	/	79	4F	117	O	O	111	6F	157	o	o
16	10	020	DLE (data link escape)	48	30	060	0	0	80	50	120	P	P	112	70	160	p	p
17	11	021	DC1 (device control 1)	49	31	061	1	1	81	51	121	Q	Q	113	71	161	q	q
18	12	022	DC2 (device control 2)	50	32	062	2	2	82	52	122	R	R	114	72	162	r	r
19	13	023	DC3 (device control 3)	51	33	063	3	3	83	53	123	S	S	115	73	163	s	s
20	14	024	DC4 (device control 4)	52	34	064	4	4	84	54	124	T	T	116	74	164	t	t
21	15	025	NAK (negative acknowledge)	53	35	065	5	5	85	55	125	U	U	117	75	165	u	u
22	16	026	SYN (synchronous idle)	54	36	066	6	6	86	56	126	V	V	118	76	166	v	v
23	17	027	ETB (end of trans. block)	55	37	067	7	7	87	57	127	W	W	119	77	167	w	w
24	18	030	CAN (cancel)	56	38	070	8	8	88	58	130	X	X	120	78	170	x	x
25	19	031	EM (end of medium)	57	39	071	9	9	89	59	131	Y	Y	121	79	171	y	y
26	1A	032	SUB (substitute)	58	3A	072	:	:	90	5A	132	Z	Z	122	7A	172	z	z
27	1B	033	ESC (escape)	59	3B	073	;	;	91	5B	133	[	[	123	7B	173	{	{
28	1C	034	FS (file separator)	60	3C	074	<	<	92	5C	134	\	\	124	7C	174			|
29	1D	035	GS (group separator)	61	3D	075	=	=	93	5D	135	]	]	125	7D	175	}	}
30	1E	036	RS (record separator)	62	3E	076	>	>	94	5E	136	^	^	126	7E	176	~	~
31	1F	037	US (unit separator)	63	3F	077	?	?	95	5F	137	_	_	127	7F	177	&#127;	DEL

Exercises:

Q1. Which of the following names are legal ?

a. ?1

b. T =

c. LET'S_GO

Q2. If it is legal, give data definition pseudo-op to define each of the following.

a. A word variable A initialized to 52.

b. A byte variable C initialized to 300.

c. A word variable WORD1,unintialized.

d. A byte variable B initialized to -129.