Data definition and memory distribution

Variable can be viewed in any numbering system:

HEX - hexadecimal (base 16).

BIN - binary (base 2).

OCT - octal (base 8).

SIGNED - signed decimal (base 10).

UNSIGNED - unsigned decimal (base 10).

CHAR - ASCII char code (there are 256 symbols, some symbols are invisible).

BCD packed and unpacked

DB define a byte
DW define a word
DD define a double word

X DB 104,-1
Y DW 100, 200H

0	X	68	
1		FF	
2	Y	64	I word
3		00	
4		00	II word
5		02	

DATA DB 3*20, -1, 100 DUP(?),?
Packed DB 78H,56H
Unpacked DB 7H,8H,5H,6H
There are 2 type of data definition: digital and addressable.

Myseg segment

X DB 0FFH; one byte equal to FF

Y DW 1234H; one word equal to 1234

Z DW Z; one word = 0003

Var DW Var+5; one word = 000A (the offset of the variable +5)

Ate DB 5*6; one byte =1E

Ss DW ?; one word without initialisation

Myseg ends

myseg	7 0	
X	FF	0
Y	34	1
	12	2
Z	03	3
	00	4
Var	0A	5
	00	6
ate	1E	7
ss	XX	8
	XX	9

Characters string definition:
Each character is stored in one byte.
The address of the string is the address of the smaller byte.

Message DB ‘HELLO’ (ASCII code in memory H(48), E(45), L(4C), O(4F)

Block DB 128(‘ ’) 128 spaces

Segment definition

- We can define data segment ,code segment and stack segment.
- In data segment variables are defined and in code segment instructions. Also in code segments can be defined the correspondence between segment and segment registers with directive **Assume.**
- **Assume CS:code_segment, DS:data_segment_1,ES:data_segment_2** it means that the segment address of Code_segment should be in CS (automatically) and data_seg in DS.

```
Name        Example    ;  the name of the assembler object module
Data_seg  segment
X db 12
Y db 230
Z db ?
Data_seg ends
Stack       segment
DW  10 DUP (?) ; reserved 10 arbitrary words for stack
Stack      ends
Code_seg segment
Assume CS:code_seg, DS: data_seg, SS: Stack
Start:        mov ax, data_seg
              mov ds, ax
              mov AX,Stack
              mov SS,,ax
              mov SP,OFFSET TOS (the 11th word)
  prog:     mov ax,0 ; ax=0
              mov al, X ;al=0Ch
              add al,Y  ;al=0ch+E6=0F2h
              mov Z,al
code_seg ends
end Start
```

We can define segments and in other way: (I column in the table)

Example of exe program:	Example of com program

TITLE Exe
.MODEL SMALL
.STACK 100h
.DATA
X DB 1,1,1,3
Y DW 2,2,2,4
.CODE
start: mov ax,@data
mov ds,ax
mov bx,1
.....
mov ax,4C00h; return control to the operating system (stop program).
int 21h
END start
	TITLE com
	prog SEGMENT
	ASSUME cs:prog,ds:prog,ss:prog
	ORG 100h
	start: jmp M1
	X DB
	Y DB .....
	M1: mov bx,1
	.....add ax,bx.....
	prog ENDS
	END start

Example with different addressing modes

```
        TITLE Adderssing modes
      .MODEL SMALL
      .STACK 100h
      .DATA
  X    DW   2 DUP(?), 34H
  Y    DW 5514H
  Z    DW 1CH
      .CODE
start: mov   ax,@DATA     ; ds=to the base address of
                          ;data segment  ds=0720H
      mov   ds,ax
      mov   ax,25          ; Imediat addr. Mode AX=0019H
```

```
mov   ax,1101101b    ; AX=006DH
mov   ax,11h         ; AX=11
mov   X,ax           ; Direct addr. Mode 0000=11H
mov   bx,Y           ;BX=14h
xchg  ax,bx          ; Interchange contents of ax
                     ; and bx AX=14H BX=11H
xchg  ax,Z           ; AX=1CH and
lea   BX, Y          ; get address of Y in BX. BX=0006
mov   bx, OFFSET Y           ; get address of Y in BX.
mov   word PTR [BX], 0AAh    ; modify the content of first
                              ;word of Y.  0006
mov  si,2
```

```
mov   X[si],ax       ; Indirect addressing mode the second
                     ;word of X is equal  to ax
lea   bx,X           ; the offset of X is loaded in BX 0000
mov   cx,[bx][si]    ; Based-indexed addr mode BX=0,SI=2
                     ;CX=1C(the  second  word of X)
mov   bx,4
mov   cx,[bx]        ;In CX is loaded the third word of X
                     ;CX=34
mov   ax,X[4]        ; The same
mov   ax,[X+4]       ; the same
mov   ax,Y           ;  AX=14
mov   di,6
mov   bx,[di]        ; BX=14
```

```
mov   BYTE PTR [di],15h; only one byte of X is
                        ;addresed   (the 7th)
 mov   BYTE PTR X+4,0BBh; The same the 5th
                         ; byte =BB
 mov   si,1
 mov   bx,3
 mov   X[bx][si],11AAh     ; Base-indexed +disp
                            ;addr.mode
                           ;The 5th byte=AA  and 6th
                           ;byte=11

  mov   X[bx+si],11AAh     ; the same
  mov   [X+bx+si],11AAh   ; the same
  mov   [bx][si].X,11AAh
  mov   [bx][si]+X,11AAh
  END  start
```

Examples

1. Define a set of 3 bytes in memory. Move data circulary, using direct addressing mode

```
.model small
.data
x db 32h, 43h, 54h
.code
 start: mov ax,@data
        mov ds,ax
        mov bl,x
        mov bh,[x+1] direct
        mov dl,[x+2]
        mov x,bh
        mov [x+1],dl
        mov [x+2],bl
        end start
```

3. Permute 5 times variables in memory

```
.model small
.data
x db 32h, 43h, 54h
.code
 start: mov ax,@data
        mov ds,ax
        mov cx,5
 L1:  mov bl,x
        mov bh,[x+1]
        mov dl,[x+2]
        mov x,bh
        mov [x+1],dl
        mov [x+2],bl
        loop L1
        end start
```

4. Load in a variable in memory its address as segment:offset

```
.model small
.data
x dw 1Ah, 1Bh, 1Ch
y  dd ?
.code
 start: mov ax,@data
        mov ds,ax
        mov bx, offset y
        mov cx, seg y
        mov word ptr y, bx
        mov word ptr y+2, cx
        end start
```

5. Define a set of 5 words in memory. Write the value 0FF in high byte of the words.

```
.model small
.data
x dw 5 dup(1122h)
num equ 0ffh
.code
start: mov ax, @data
       mov ds,ax
       mov al, num
       mov si,1
       mov cx,5
label: mov byte ptr x[si],al
       add si, 2
       loop label
   end start
```

Laboratory work. Addressing modes

- Declare memory spaces for 3 variables (DB, DW and DD). Use immediate addressing mode to load these variables in general purpose registers.
- Store the content of the used registers in stack, reset registers with ***mov*** instruction and then restore the content of the registers from stack.
- Use as many memory addressing modes as is possible to store the content of the registers in memory and vice-versa. Consider DS=ADDRESS. Reset each time registers with ***xor*** instruction.

No	DS	No	DS
1	30A4H	15	4210H
2.	A234H	16.	6666H
3.	E987H	17.	CD34H
4.	123FH	18.	23DDH
5.	54AAH	19.	9999H
6.	76BBH	20.	543AH
7.	3A06H	21.	EEAAH
8.	23BBH	22.	1234H
9.	4210H	23.	B234H
10.	6666H	24.	EA87H
11.	CD34H	25.	1C3FH
12.	23DDH	26.	5FAAH
13.	9999H	27.	D6BBH
14.	543AH	28.	1A06H