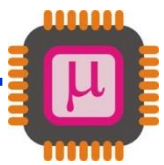


Fall 2019/20 – Lecture Notes # 6

- Directives and sample programs
- Full Segment Definition
- Simplified Segment Definition
- Assemble, Link and run a program



Directives and sample programs

• Directives

- **Directives** are statements that give directions to the assembler about how it should translate the assembly language instructions into machine code.
- An assembly language instruction consists of four fields,

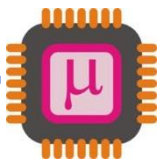
[label:] mnemonic [operands] [;comments]

1. Brackets indicate that the field is optional. Brackets are not typed.
2. The **label** field allows the program to refer to a line of code by name. In a line of assembly language program there can be mnemonic (instruction) and operand(s).

Example :

ADD	AL,BL
MOV	AX,6764H

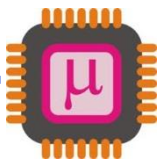
3. Alternatively, instead of these two fields there can be directives. Directives are used by the assembler to organize the program as well as other output files. The following program adds two bytes to calculate their sum. IN this program SEGMENT, DB, ENDS, ASSUME, END, and ENDP are examples of directives.
4. The comment field begins with a “;”



Directives and Sample Program

;A Sample Assembly Language Program using **FULL SEGMENT DEFINITION**

```
STSEG      SEGMENT
            DB 64 DUP (?)
STSEG      ENDS
;-----
DTSEG      SEGMENT
DATA1      DB      52H
DATA2      DB      29H
SUM        DB      ?
DTSEG      ENDS
;-----
CDSEG      SEGMENT
MAIN       PROC    FAR                ;This is the program entry point
            ASSUME CS:CDSEG,DS:DTSEG,SS:STSEG
            MOV    AX,DTSEG           ;load the data segment address
            MOV    DS,AX             ;assign value to DS
            MOV    AL,DATA1         ;get the first operand
            MOV    BL,DATA2         ;get the second operand
            ADD    AL,BL             ;add the operands
            MOV    SUM,AL           ;store result in location SUM
            MOV    AH,4CH            ;set up to
            INT    21H              ;return to the Operating System (DOS)
MAIN       ENDP
CDSEG      ENDS
            END        MAIN         ;this is the program exit point
```



Directives and Sample Program

- **Program segments:**

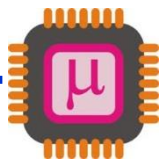
```
label    SEGMENT [options]
          ;place the statements belonging to this segment here
label    ENDS
```

The stack segment defines storage for the stack, the data segment defines the data that the program will use and the code segment contains the Assembly Language instructions.

- **Stack segment definition**

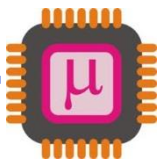
```
STSEG    SEGMENT                ;the "SEGMENT" directive begins the segment
          DB 64 DUP (?)         ;this segment contains only one line
STSEG    ENDS                   ;the "ENDS" segment ends the segment
```

DB 64 DUP (?) , directive reserves 64 bytes of memory for the stack.



Directives and Sample Program

- **Data segment definition**
 - There are three data items in this sample program: DATA1, DATA2 and SUM. Each is defined as **DB (define byte)**.
 - The **DB** directive is used to allocate memory in byte-sized chunks. **DW (define word)** allocates 2 bytes of memory. DATA1 and DATA2 have initial values but SUM is reserved for later use.
- **Code segment definition**
 - The first line after the **SEGMENT directive** is the **PROC directive**. A procedure is a group of instructions designed to accomplish a specific function.
 - The **PROC and ENDP** directives must have the same label. The PROC directive may have the option **FAR or NEAR**. DOS requires FAR option to be used at the program entry.
 - ASSUME directive associates segment registers with specific segments by assuming that the segment register is equal to the segment labels used in the program.
 - Note that there can be many segments of the same type. So Assume helps to differentiate which is to be used at a time.



Directives and Sample Program

- **Code segment definition**

- OS determines the CS and SS segment registers automatically. DS has to be manually specified.

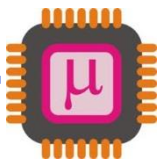
```
MOV    AX,DTSEG    ;load the data segment address
MOV    DS,AX       ;assign value to DS
```

- Load AL and BL with DATA1 and DATA2 and ADD them together, and store the result in SUM.

```
MOV    AL,DATA1    ;get the first operand
MOV    BL,DATA2    ;get the second operand
ADD    AL,BL       ;add the operands
MOV    SUM,AL      ;store result in location SUM
```

- The last two instructions returns the control to the operating system.

```
MOV    AH,4CH      ;set up to
INT    21H         ;return to DOS
```



Directives and Sample Program

- **Simplified Segment Definition**

- An assembly language program can be written in two segment definition formats. In addition to the **Full Segment Definition** Format, the recent assemblers support **Simplified Segment Definition**, which is simpler and easier to write.

;A Sample Assembly Language Program using **SIMPLIFIED SEGMENT DEFINITION**

```
.MODEL SMALL           ;Gives the memory model to be used by the program
```

```
.STACK 64
```

```
;-----
```

```
.DATA
```

```
DATA1      DB      52H
```

```
DATA2      DB      29H
```

```
SUM        DB      ?
```

```
;-----
```

```
.CODE
```

```
MAIN:      MOV AX,@DATA
```

```
MOV DS,AX      ;assign value to DS
```

```
MOV AL,DATA1 ;get the first operand
```

```
MOV BL,DATA2 ;get the second operand
```

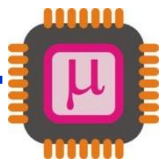
```
ADD AL,BL    ;add the operands
```

```
MOV SUM,AL   ;store result in location SUM
```

```
MOV AH,4CH     ;set up to
```

```
INT 21H        ;return to the Operating System (DOS)
```

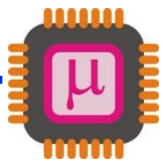
```
END MAIN       ;this is the program exit point
```



Directives and Sample Program

- **Simplified Segment Definition**

- In a program with Simplified Segment Definition, the memory model that the program will use must be specified. The following **Memory Models** can be used:
 - **SMALL MODEL (.MODEL SMALL):** The model uses maximum of **64K bytes for Code** and **64K bytes for Data (Code≤64K and Data ≤64K)**. This model is the most widely used memory model and is sufficient for all the programs to be used in this course.
 - **MEDIUM MODEL, (.MODEL MEDIUM):** The model uses maximum of **64K bytes for Data** and Code can exceed **64K bytes (Code>64K and Data ≤64K)**.
 - **COMPACT MODEL, (.MODEL COMPACT):** The model uses maximum of **64K bytes for Code** and Data can exceed **64K bytes (Code≤64K and Data >64K)**.
 - **LARGE MODEL, (.MODEL LARGE):** Both Code and Data can exceed **64K bytes. However no single data set (i.e. array) can exceed 64K bytes (Code>64K and Data >64K)**.
 - **HUGE MODEL, (.MODEL HUGE):** Both Code and Data can exceed **64K bytes**. Additionally, a single data set (i.e. array) can exceed 64K bytes (**Code>64K and Data >64K**).

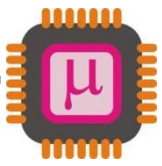


Directives and Sample Program

- **Assemble, Link and run a program**
 - The following table summarizes the process required to assemble, link and run an assembly language program.

Step	Input	Program	Output
1. Edit the program	keyboard	Editor	myfile.asm
2. Assemble the program	myfile.asm	MASM or TASM	myfile.obj
3. Link the program	myfile.obj	LINK or TLINK	myfile.exe

- “.asm” file is the source file created with an editor or a word processor.
- “.obj” assembler (e.g.TASM) converts .asm file’s Assembly language instructions into machine language.
- “.exe” TLINK is the program to produce the executable file.



Directives and Sample Program

- **Assemble, Link and run a program**
 - **DEBUG**, is a program included in DOS operating system that allows the programmer to monitor the programs execution.
 - **Useful commands are:**
 - u (unassemble command is used to look et the machine code)
 - d (dump command displays the contents of memory to the screen)
 - g (go command executes the program)
 - q (quits from DEBUG to DOS)