Data Type in assembly language

The microprocessor has multiple data type formats like binary, BCD, ASCII, signed and unsigned numbers.

Data Type	Description
DB	Define Byte (Size – 1 Byte)
DW	Define Word (Size – 2 Byte)
DD	Define Double word (Size - 4 Bytes)
DQ	Define Quad word (Size – 8 Bytes)
DT	Define Ten Bytes (Size – 10 Bytes)

The 8086 microprocessor supports 8 types of instructions –

- Data Transfer Instructions
- Arithmetic Instructions
- Bit Manipulation Instructions
- String Instructions
- Program Execution Transfer Instructions (Branch & Loop Instructions)
- Processor Control Instructions
- Iteration Control Instructions
- Interrupt Instructions

Let us now discuss these instruction sets in detail.

Data Transfer Instructions

These instructions are used to transfer the data from the source operand to the destination operand. Some instruction have one operand and some has two operands .Following are the list of instructions under this group –

Such as

INC, DEC, MUL, PUSH and POP etc. has one operand

MOV, ADD, SUB and XCHG etc. have two operands.

Instruction to transfer a word

- **MOV** – Used to copy the byte or word from the provided source to the provided destination.
- **PUSH** – Used to put a word at the top of the stack.
- **POP** – Used to get a word from the top of the stack to the provided location.
- **PUSHA** – Used to put all the registers into the stack.
- **POPA** – Used to get words from the stack to all registers.

- **XCHG** – Used to exchange the data from two locations.
- **XLAT** – Used to translate a byte in AL using a table in the memory.

Instructions for input and output port transfer

- **IN** – Used to read a byte or word from the provided port to the accumulator.
- **OUT** – Used to send out a byte or word from the accumulator to the provided port.

Instructions to transfer the address

- **LEA** – Used to load the address of operand into the provided register.
- **LDS** – Used to load DS register and other provided register from the memory
- **LES** – Used to load ES register and other provided register from the memory.

Instructions to transfer flag registers

- **LAHF** – Used to load AH with the low byte of the flag register.
- **SAHF** – Used to store AH register to low byte of the flag register.
- **PUSHF** – Used to copy the flag register at the top of the stack.
- **POPF** – Used to copy a word at the top of the stack to the flag register.

Arithmetic Instructions

These instructions are used to perform arithmetic operations like addition, subtraction, multiplication, division, etc.

Following is the list of instructions under this group –

Instructions to perform addition

- **ADD** – Used to add the provided byte to byte/word to word.
- **ADC** – Used to add with carry.
- **INC** – Used to increment the provided byte/word by 1.
- **AAA** – Used to adjust ASCII after addition.
- **DAA** – Used to adjust the decimal after the addition/subtraction operation.

Instructions to perform subtraction

- **SUB** – Used to subtract the byte from byte/word from word.
- **SBB** – Used to perform subtraction with borrow.
- **DEC** – Used to decrement the provided byte/word by 1.
- **NPG** – Used to negate each bit of the provided byte/word and add 1/2's complement.
- **CMP** – Used to compare 2 provided byte/word.
- **AAS** – Used to adjust ASCII codes after subtraction.

- **DAS** – Used to adjust decimal after subtraction.

Instruction to perform multiplication

- **MUL** – Used to multiply unsigned byte by byte/word by word.
- **IMUL** – Used to multiply signed byte by byte/word by word.
- **AAM** – Used to adjust ASCII codes after multiplication.

Instructions to perform division

- **DIV** – Used to divide the unsigned word by byte or unsigned double word by word.
- **IDIV** – Used to divide the signed word by byte or signed double word by word.
- **AAD** – Used to adjust ASCII codes after division.
- **CBW** – Used to fill the upper byte of the word with the copies of sign bit of the lower byte.
- **CWD** – Used to fill the upper word of the double word with the sign bit of the lower word.

Bit Manipulation Instructions

These instructions are used to perform operations where data bits are involved, i.e. operations like logical, shift, etc.

Following is the list of instructions under this group –

Instructions to perform logical operation

- **NOT** – Used to invert each bit of a byte or word.
- **AND** – Used for adding each bit in a byte/word with the corresponding bit in another byte/word.
- **OR** – Used to multiply each bit in a byte/word with the corresponding bit in another byte/word.
- **XOR** – Used to perform Exclusive-OR operation over each bit in a byte/word with the corresponding bit in another byte/word.
- **TEST** – Used to add operands to update flags, without affecting operands.

Instructions to perform shift operations

- **SHL/SAL** – Used to shift bits of a byte/word towards left and put zero(S) in LSBs.
- **SHR** – Used to shift bits of a byte/word towards the right and put zero(S) in MSBs.
- **SAR** – Used to shift bits of a byte/word towards the right and copy the old MSB into the new MSB.

Instructions to perform rotate operations

- **ROL** – Used to rotate bits of byte/word towards the left, i.e. MSB to LSB and to Carry Flag [CF].
- **ROR** – Used to rotate bits of byte/word towards the right, i.e. LSB to MSB and to Carry Flag [CF].
- **RCR** – Used to rotate bits of byte/word towards the right, i.e. LSB to CF and CF to MSB.
- **RCL** – Used to rotate bits of byte/word towards the left, i.e. MSB to CF and CF to LSB.

String Instructions

String is a group of bytes/words and their memory is always allocated in a sequential order.

Following is the list of instructions under this group –

- **REP** – Used to repeat the given instruction till CX ≠ 0.
- **REPE/REPZ** – Used to repeat the given instruction until CX = 0 or zero flag ZF = 1.
- **REPNE/REPNZ** – Used to repeat the given instruction until CX = 0 or zero flag ZF = 1.
- **MOVS/MOVSB/MOVSW** – Used to move the byte/word from one string to another.
- **COMS/COMPSB/COMPSW** – Used to compare two string bytes/words.
- **INS/INSB/INSW** – Used as an input string/byte/word from the I/O port to the provided memory location.
- **OUTS/OUTSB/OUTSW** – Used as an output string/byte/word from the provided memory location to the I/O port.
- **SCAS/SCASB/SCASW** – Used to scan a string and compare its byte with a byte in AL or string word with a word in AX.
- **LODS/LODSB/LODSW** – Used to store the string byte into AL or string word into AX.

Program Execution Transfer Instructions (Branch and Loop Instructions)

These instructions are used to transfer/branch the instructions during an execution. It includes the following instructions –

Instructions to transfer the instruction during an execution without any condition –

- **CALL** – Used to call a procedure and save their return address to the stack.
- **RET** – Used to return from the procedure to the main program.
- **JMP** – Used to jump to the provided address to proceed to the next instruction.

Instructions to transfer the instruction during an execution with some conditions –

- **JA/JNBE** – Used to jump if above/not below/equal instruction satisfies.
- **JAE/JNB** – Used to jump if above/not below instruction satisfies.
- **JBE/JNA** – Used to jump if below/equal/ not above instruction satisfies.

- **JC** – Used to jump if carry flag CF = 1
- **JE/JZ** – Used to jump if equal/zero flag ZF = 1
- **JG/JNLE** – Used to jump if greater/not less than/equal instruction satisfies.
- **JGE/JNL** – Used to jump if greater than/equal/not less than instruction satisfies.
- **JL/JNGE** – Used to jump if less than/not greater than/equal instruction satisfies.
- **JLE/JNG** – Used to jump if less than/equal/if not greater than instruction satisfies.
- **JNC** – Used to jump if no carry flag (CF = 0)
- **JNE/JNZ** – Used to jump if not equal/zero flag ZF = 0
- **JNO** – Used to jump if no overflow flag OF = 0
- **JNP/JPO** – Used to jump if not parity/parity odd PF = 0
- **JNS** – Used to jump if not sign SF = 0
- **JO** – Used to jump if overflow flag OF = 1
- **JP/JPE** – Used to jump if parity/parity even PF = 1
- **JS** – Used to jump if sign flag SF = 1

Processor Control Instructions

These instructions are used to control the processor action by setting/resetting the flag values.

Following are the instructions under this group –

- **STC** – Used to set carry flag CF to 1
- **CLC** – Used to clear/reset carry flag CF to 0
- **CMC** – Used to put complement at the state of carry flag CF.
- **STD** – Used to set the direction flag DF to 1
- **CLD** – Used to clear/reset the direction flag DF to 0
- **STI** – Used to set the interrupt enable flag to 1, i.e., enable INTR input.
- **CLI** – Used to clear the interrupt enable flag to 0, i.e., disable INTR input.

Iteration Control Instructions

These instructions are used to execute the given instructions for number of times. Following is the list of instructions under this group –

- **LOOP** – Used to loop a group of instructions until the condition satisfies, i.e., CX = 0
- **LOOPE/LOOPZ** – Used to loop a group of instructions till it satisfies ZF = 1 & CX = 0
- **LOOPNE/LOOPNZ** – Used to loop a group of instructions till it satisfies ZF = 0 & CX = 0
- **JCXZ** – Used to jump to the provided address if CX = 0

Interrupt Instructions

These instructions are used to call the interrupt during program execution.

- **INT** – Used to interrupt the program during execution and calling service specified.
- **INTO** – Used to interrupt the program during execution if OF = 1
- **IRET** – Used to return from interrupt service to the main program

Structure of Assembly Program.

.model small ; decide programming memory size

.stack 100h ; decide stack size in the memory segment

.data ; Here decide data segment in the memory

Declare variables

.code ; here code segment

Main proc ; main procedure start from here

… assembly code
.. assembly code
...assembly code
Main endp ; end main procedure here

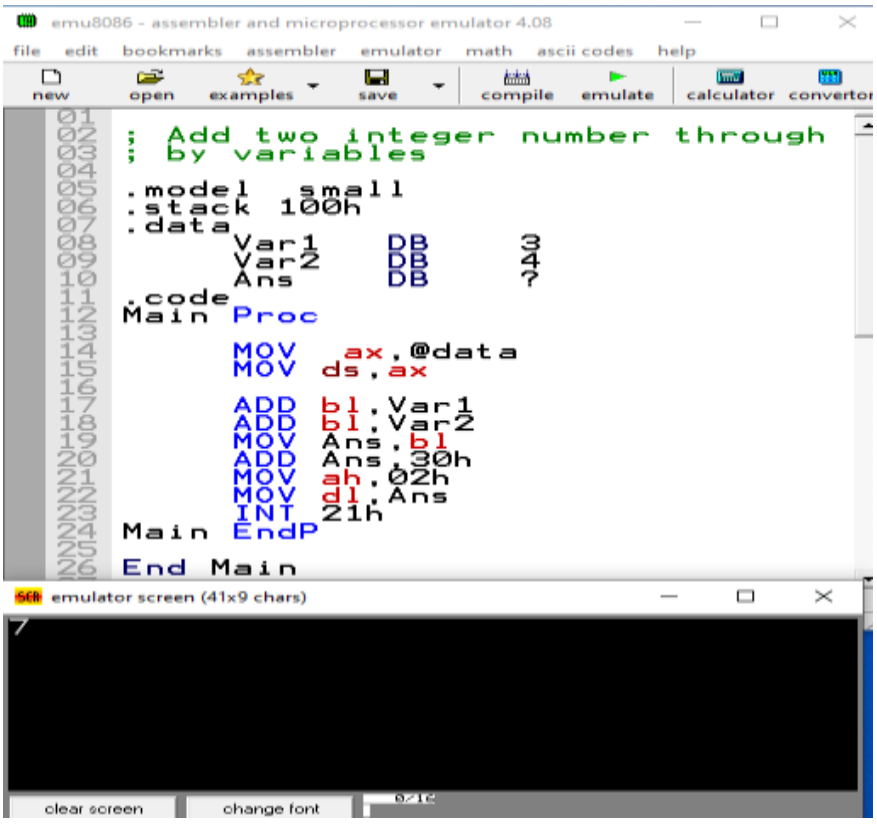
Here could be another procedure

End main

Example: Add two-integer number through by variables.

```
Example
.model  small
.stack 100h
.data
   Var1   DB    3
   Var2   DB    4
   Ans     DB    ?
.code
Main Proc
        MOV  ax,@data
        MOV ds,ax

        ADD bl,Var1
        ADD bl,Var2
        MOV Ans,bl
        ADD Ans,30h
        MOV ah,02h
        Mov dl,Ans
        INT 21h
Main EndP
End Main
```



Exercise

Theory Questions.

1. Define data types of assembly language.
2. Write structure of assembly program with description.
3. How many type instruction of 8086 microprocessor in the assembly language.

Practical Questions.

1. Write assembly code to display your name with proper structure of assembly program and declare variable for your name.

Objective and MCQs:

1. DF It uses for occupied _____ bytes.
   a) 1
   b) 2.
   c) 30
   d) 4

2. Define Double word for which data type_____.
   a) DB
   b) DW
   c) DD
   d) DQ

3. _____ Used to exchange the data from two locations not.
   a) MOV
   b) ADD
   c) XCHG
   d) LOOP

4. DIV command have _____ operand's
   a) 1.
   b) 2.
   c) 3.
   d) Not in all.

5. ____ Used to call a procedure and save their return address to the stack.
   a) JUMP
   b) JE
   c) LOOP
   d) CALL

6. LOOP Used to loop a group of instructions until the condition satisfies, i.e., CX = ?.
   a) 10
   b) 5
   c) 1
   d) 0
7. AND command is a _____ type of instruction operation.
   a) Data transfer
   b) Arithmetical
   c) logical
   d) Both a and c

8. _____ Used to interrupt the program during execution and calling service specified.
   a) LOOP
   b) JUMP.
   c) CALL.
   d) INT.

9. JNB and JAE both are perform equal operation.
   a) true
   b) false

10. ____ instruction is Used to jump if sign flag SF = 1
    a) JNE
    b) JS
    c) JE
    d) JNE