**Laboratory Session-2**

**Write-up on Instruction group, Timing diagrams.**

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

**Data Transfer Instructions**

These instructions are used to transfer the data from the source operand to the destination operand. Following are the list of instructions under this group −

**Instruction to transfer a word**

* **MOV** − Used to copy the byte or word from the provided source to the provided destination.
* **PPUSH** − 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

**Timing diagrams**

**Basic Bus Operation**

If data are written to the memory ,the microprocessor outputs the memory address on the address bus, outputs the data to be written into memory on the data bus, and issues a write (WR ) to memory and IO/M= 0 for the 8088 and M/IO= 1 for the 8086.

If data are read from the memory, the microprocessor outputs the memory address on the address bus, issues a read memory signal (RD ), and accepts the data via the data bus.

**Timing in General**

The 8086/8088 microprocessors use the memory and I/O in periods called bus cycles.

* Each bus cycle equals four system-clocking periods (T states).
* If the clock is operated at 5 MHz , one 8086/8088 bus cycle is complete in 800 ns.
* This means that the microprocessor reads or writes data between itself and memory or I/O at a maximum rate of 1.25 million times a second.

**During T1 : First Clock Period**

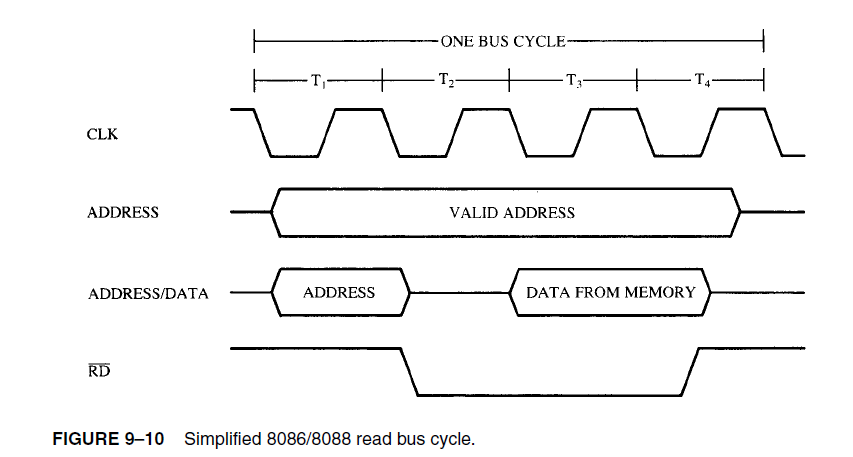
* The address of the memory or I/O location is sent out via the address bus and the address/data bus connections.
* control signals ALE, DT/R , and IO/M (8088) or M/IO (8086) are also output. The IO/M or M/IO signal indicates whether the address bus contain a memory address or an I/O device (port) number

**During T2**

The 8086/8088 microprocessors issue the RD or WR signal, DEN , and in the case of a write, the data to be written appear on the data bus.

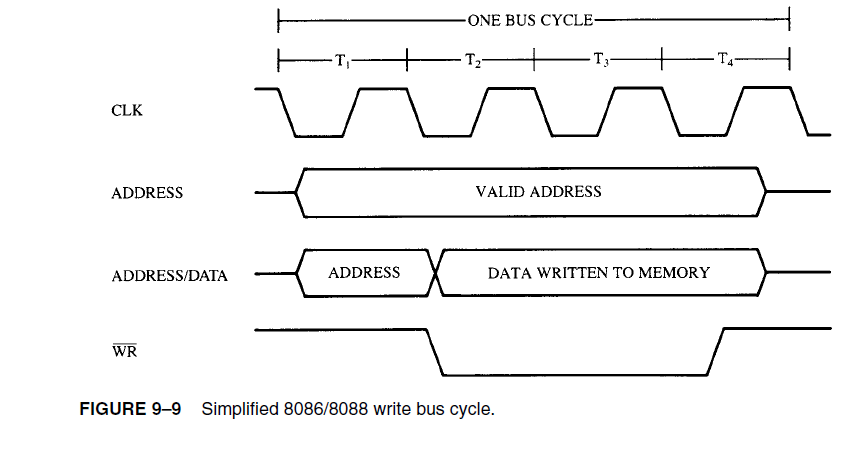
* These events cause the memory or I/O device to begin to perform a read or a write.
* The DEN signal turns on the data bus buffers, if they are present in the system, so the memory or I/O can receive data to be written, or so the microprocessor can accept the data read from the memory or I/O for a read operation.
* If this happens to be a write bus cycle, the data are sent out to the memory or I/O through the data bus.
* READY is sampled at the end of T2, If READY is low at this time, T3 becomes a wait state (Tw).
* This clocking period is provided to allow the memory time to access data. If the bus cycle happens to be a read bus cycle, the data bus is sampled at the end of T3.
* In T4, all bus signals are deactivated in preparation for the next bus cycle.
* This is also the time when the 8086/8088 samples the data bus connections for data that are read from memory or I/O.
* In addition, at this point, the trailing edge of the WR signal transfers data to the memory or I/O, which activates and writes when the WR signal returns to a logic 1 level.

**Read Timing**



* The most important item contained in the read timing diagram is the amount of time allowed for the memory or I/O to read the data.
* Memory is chosen by its access time, which is the fixed amount of time that the microprocessor allows it to access data for the read operation.
* It is therefore extremely important that the memory chosen complies with the limitations of the system.
* To find memory access time in this diagram, first locate the point in T3 when data are sampled. If you examine the timing diagram closely, you will notice a line that extends from the end of T3 down to the data bus. At the end of T3, the microprocessor samples the data bus.
* Memory access time starts when the address appears on the memory address bus and continues until the microprocessor samples the memory data at T3.
* Approximately three T states elapse between these times. (See Figure 9–12 for the following times.) The address does not appear until TCLAV time (110 ns if the clock is 5 MHz) after the start of T1.

**Write Timing**



1. When processor is ready to initiate the bus cycle, it applies a pulse to ALE during T1. Before the falling edge of ALE, the address, BHE, M/IO, DEN and DT/R must be stable i.e. DEN = high and DT/R = 0 for input or DT/R = 1 for output.
2. At the trailing edge of ALE, ICs 74LS373 or 8282 latches the address.
3. During T2 the address signals are disabled and S3-S7 ale available on AD16/S3-AD19/S6 and BHE/S7. Also DEN is lowered to enable transceiver.
4. In case of input operation, RD is activated during T2 and AD° to AD15 go in high impedance preparing for input.
5. If memory or I/O interface can perform the transfer immediately; there are no wait states and data is output on the bus during T3.
6. After the data is accepted by the processor, RD is raised high at the beginning of T4.
7. Upon detecting this transition during T.4, the memory or I/O device will disable its data signals.
8. For an output operation, processor applies WR = 0 and then the data on the data bus during T2.
9. In T4, WR is raised high and data signals are disabled.
10. For either input or output operation, DEN is raised during 14 to disable the transceiver. Also M/I0 is set according to the next transfer at this time or during next T1 state. Thus length of bus cycle in 8086 is four clock cycle. If the bus is to be inactive after completion of bus cycle, then the gap between the successive cycles is filled by ideal state clock cycles.

When the memory or I/O device is not able to respond quickly during transfer, wait states (Tw) are inserted between T3 and T4 by disabling the READY input of the 8086. The bus activity during wait state is same as during T3.