UNIT-3: INSTRUCTION CYCLE

RESISTER TRANSFER LANGUAGE (RTL)

During the execution of an instruction different actions are exercised. These operations are expressed using a language. Such type of language, which is basically used to express the transfer of data among the registers, is called Register Transfer Language (RTL). If a data is transferred from register B to register A the in RTL

register A ← register B

During the execution of an instruction we have fetch cycle and execute cycle. The operations performed during fetch cycle are:

- The PC contains the address of the next instruction to be executed. As first operation of fetch cycle, the contents of program counter will be transferred to the memory address register (MAR). The memory address register then uses the address bus to transmit its contents that specifies the address of the memory location from where the instruction code is to be fetched. Let t1 be period of this operation.

  MAR ← PC
- Now as soon as the control unit issues the memory read signal, the contents of the addressed memory location specified by MAR will be transferred to the memory buffer register (MBR). Let t2 be the period of this operation.

  MBR ← Memory

- Now the contents of MBR will be transferred to the instruction register and the program counter (PC) gets incremented to fetch another instruction and the fetching cycle is thus complete. Here two operations take place within a single time unit t3. Let t3 be the time required by the CPU for this operation.

  IR ← (MBR)
  PC ← PC+1

The control unit of the computer maintains the timing sequence of the all of the above operations that constitute the fetch cycle. The operations are sequenced on the basis of the single time period called the period of the clock. The RTL for fetch cycle is:

t1: MAR ← PC
t2: MBR ← Memory
t3: IR ← (MBR)
PC ← PC +1

All three time units are of equal duration. A time unit is defined by a regularly spaced clock pulses. The operations performed within this single unit of time are called micro-operations. Since each micro-operation specifies the transfer of data into or out of a register, such type of language is called Register Transfer Language.

After the fetch of the instruction is complete, the execution cycle starts. Different instructions are executed in different fashions. Let us consider execution of a simple instruction MOV A,B. The first step needed is to obtain the data from the location B. For this the address field of instruction register indicating the address of memory location will

be transferred to address bus through the Memory Address Register (MAR).

When the control unit issues a memory read signal, the contents of location B will be output to the MBR. Now the address of memory location A is loaded in MAR. Finally, after the memory write signal from the control unit is issued the data is copied to memory location A. Using RTL language and defining above operations in the form of micro operation the execute cycle is represented in following time units.

t1: MAR ← (IR (Address of B))
t2: MBR ← (B)
t3: MAR ← (IR (Address of A))
t4: A ← MBR

INSTRUCTION AND MACHINE CYCLE

Consider a simple instruction: MOV A, B which states that data from register B is to copied to register A. The instructions are all stored in the memory. The computer spends certain period of time on Fetching, Decoding and Executing this instruction.

Instruction Cycle: It is defined as the time required to complete execution of an instruction.

Machine cycle: It is defined as the time required to complete one operation of accessing memory, I/O, or acknowledging an external request.

T-state: It is defined as one subdivision of the operation performed in one clock period.

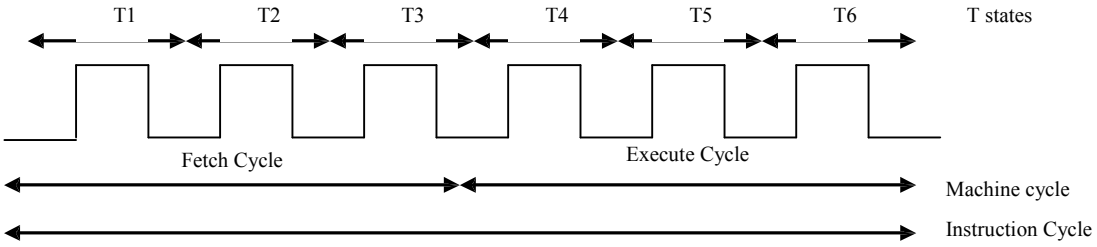


Fig Instruction cycle

Fetch and execution cycle, fetch execution overlap

The processor does the actual work by executing instructions specified in the program. In its simplest form instruction processing consists of two steps: The processor reads instructions from memory one at a time and executes each instruction. Program execution consists of repetitive fetch and instruction execution.

The processing time required for a single instruction is called an instruction cycle. The instruction cycle is composed of: fetch cycle and execute cycle. During fetch the instruction is read from its memory location into the processor. The time required for this memory read operation is called fetch cycle. After the instruction is fetched it ought to be executed. The time required for execution is the execution cycle.

The fetch and execute stages of instruction execution are quite independent. The fetch cycle involves accessing the main memory for instruction. But during execution the memory need not be referenced. So, during this duration the next instruction can be fetched. This is called instruction prefetch. This prefetching data before it is asked for in the anticipation that it will be used is called pipelining. The process of pipelining enhances the speed of processor greatly. However, 8085 doesn‟t support pipelining. Thus there is no overlap between fetch cycle and execution cycle.

2.6 Timing diagram for register move, indirect read, indirect write and out instructions

The 8085 microprocessor is designed to execute 74 different instruction types. Each instruction has two parts: operation code, known as op-code, and operand. To execute an instruction, the 8085 needs to perform various operations such as Memory Read/Write and I/O Read/ Write. The total operations of a microprocessor can be classified into the following operations.

- Op-code Fetch
- Memory Read and Write
- I/O Read and Write
- Request Acknowledge

Op-code Fetch Cycle:
The first operation in any instruction is Op-code Fetch. The microprocessor needs to get the machine code from the memory register where it is stored before the microprocessor can begin to execute any instruction.

Let us consider the timing for execution of the instruction MVI A,32H. The op-code of MVI is 3EH. Since the op-code fetch cycle is analogous for all the instructions all op-code fetch cycle consists of four machine cycles.

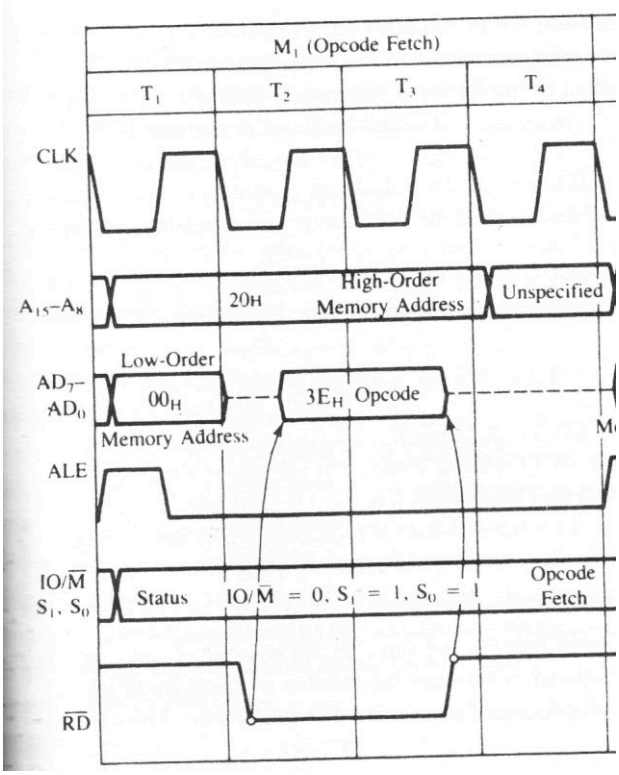


Fig: Timing Diagram of Op-code Fetch Cycle

The following figure shows the timing of how a data is transferred from memory to MPU; it has five different groups of signals in relation to system clock. The address bus and data bus are shown as two parallel lines. Other control signals are shown using single lines representing logic levels. The crossover of the lines indicates a new byte is placed on the bus. The dashed straight line indicates the high impedance state.

The following steps occur during Op-code Fetch Cycle

1. The Program counter (PC) places the 16 bit memory address on the address bus. At T1 high order address is placed at A_8-A_{15} and lower order address is placed at AD_0-AD_7 and the ALE signal goes high, IO/$\overline{M}$ goes low, and both s0 and s1 goes high; which identifies the op-code fetch cycle.
2. The control unit sends the control signal RD to enable the memory chip and remains active till two clock periods.
3. Now the op-code from memory location is placed on the multiplexed data bus. In this case 3E is placed on $AD_7 - AD_0$. The transition of RD indicates the end of this step.
4. The op-code byte is now placed on instruction decoder and the execute cycle is carried out

Memory Read cycle:

The Op-code fetch cycle is a memory read cycle. Thus, other memory read cycles are similar to the op-code fetch cycle. Let us take the same example as above. The instruction cycle of MVI A,32H is shown below:

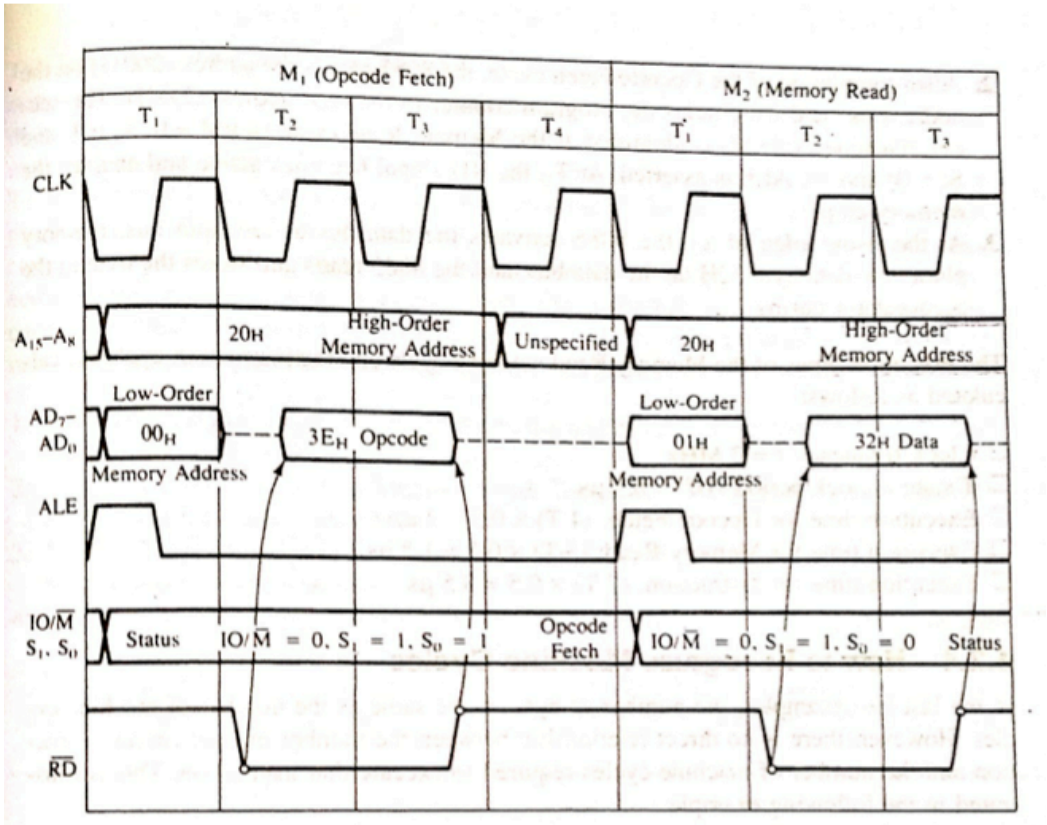


FIGURE 4.10
8085 Timing for Execution of the Instruction MVI A,32H

The total cycle consist of 7 T states and 2 machine cycles: op-code fetch and memory read. At the end of op-code fetch the PC is incremented thus the address is now 2001h and instruction decoder has 3Eh.

Now the operand is to be read from the memory to register A. The second machine cycle is the memory read and consists of 3 T states. The signal content of this cycle are similar to the first three T states of op-code fetch except the status signal, In this case s0=0 and s1=1.

The byte read in T3 of memory read cycle will be copied into the accumulator in the same cycle.

I/O Write Cycle:

Let us consider the instruction OUT 01h stored at memory location 2050h. The op-code of this instruction is D3h and the complete cycle requires 10 T states. The write cycle executes as shown below:

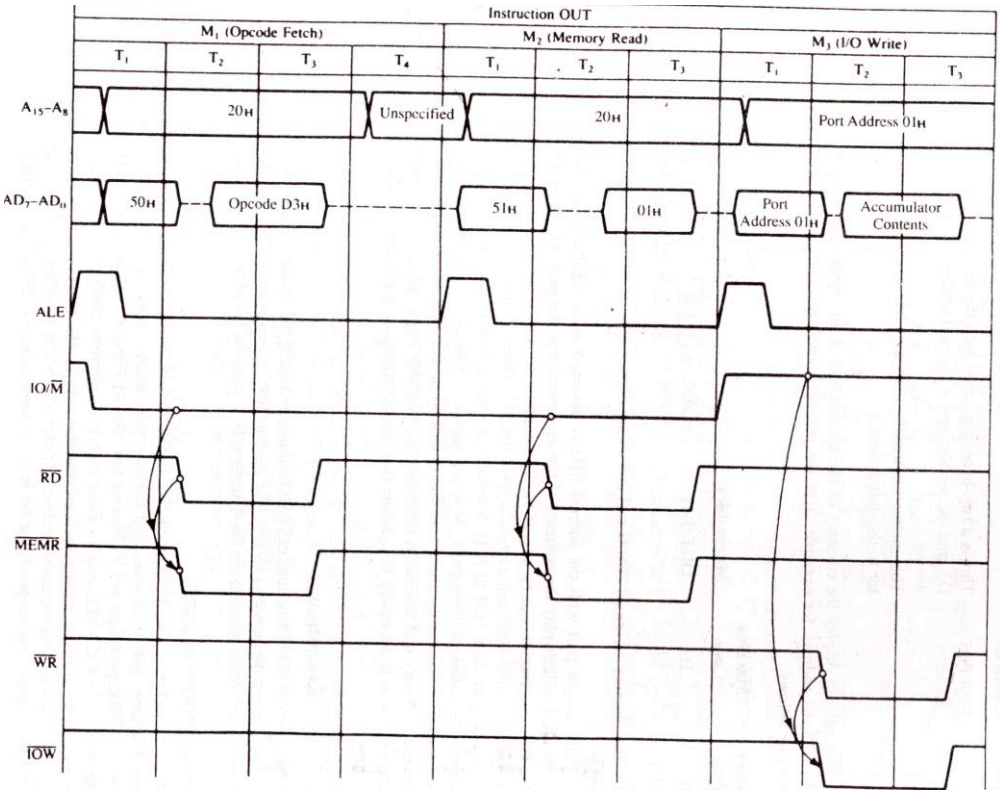


Fig: Timing diagram of I/O write cycle

The instruction is executed under three machine cycles: op-code fetch, memory read and I/O write. The op-code fetch and memory read cycles exactly match the previous ones except for the contents of the buses. At the end of memory read cycle the PC points at 2051h.

Now instruction decoder contains D3h and on the next machine cycle the port address 01h is placed on higher and lower port address. Unlike previous cases now IO/M signal goes high to indicate a I/O operation. At T2 the accumulator contents are placed on the data bus, followed by the control signal WR. The signal IOW is obtained by anding the IO/M and WR signals to enable the output device

I/O Read Cycle:

The I/O read cycle is exactly same as I/O write cycle. The only difference is on the third machine cycle where the signal IOR is generated and the content of the port is read to the accumulator instead of otherwise. The IOR signal is generated by anding the IO/M and RD signals to enable the input device.

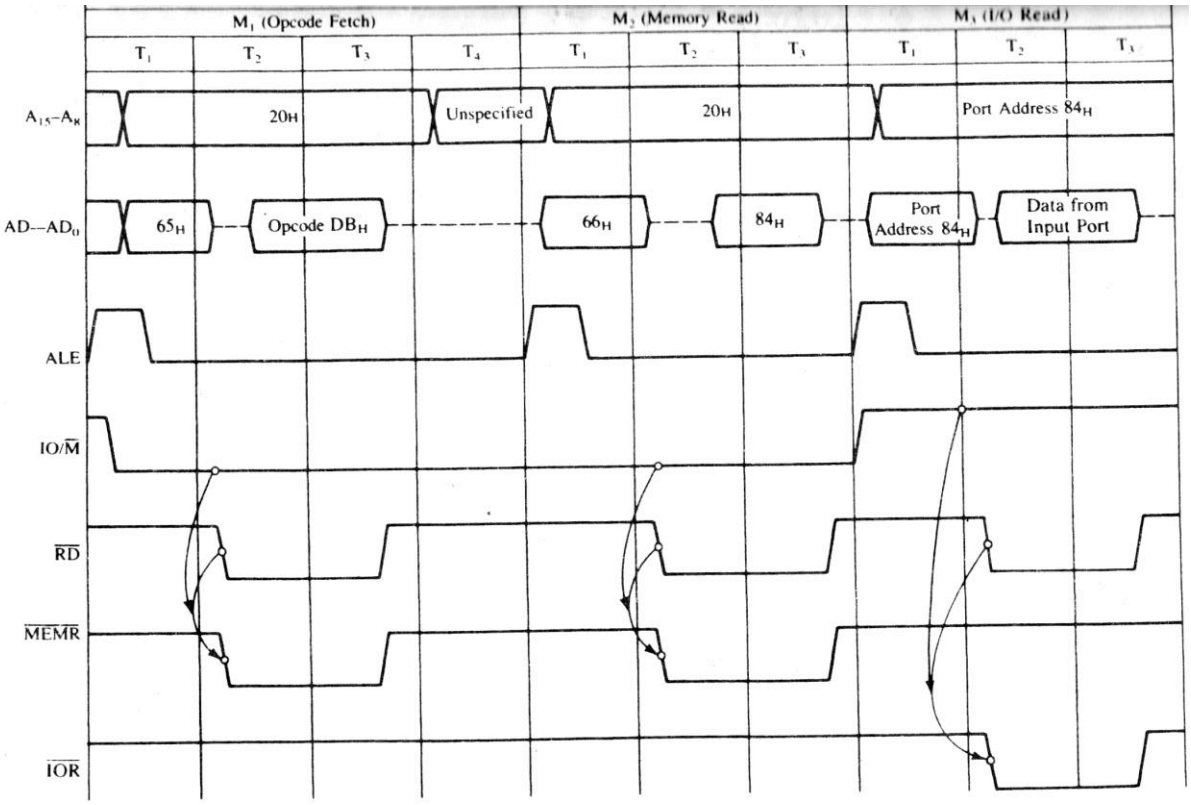
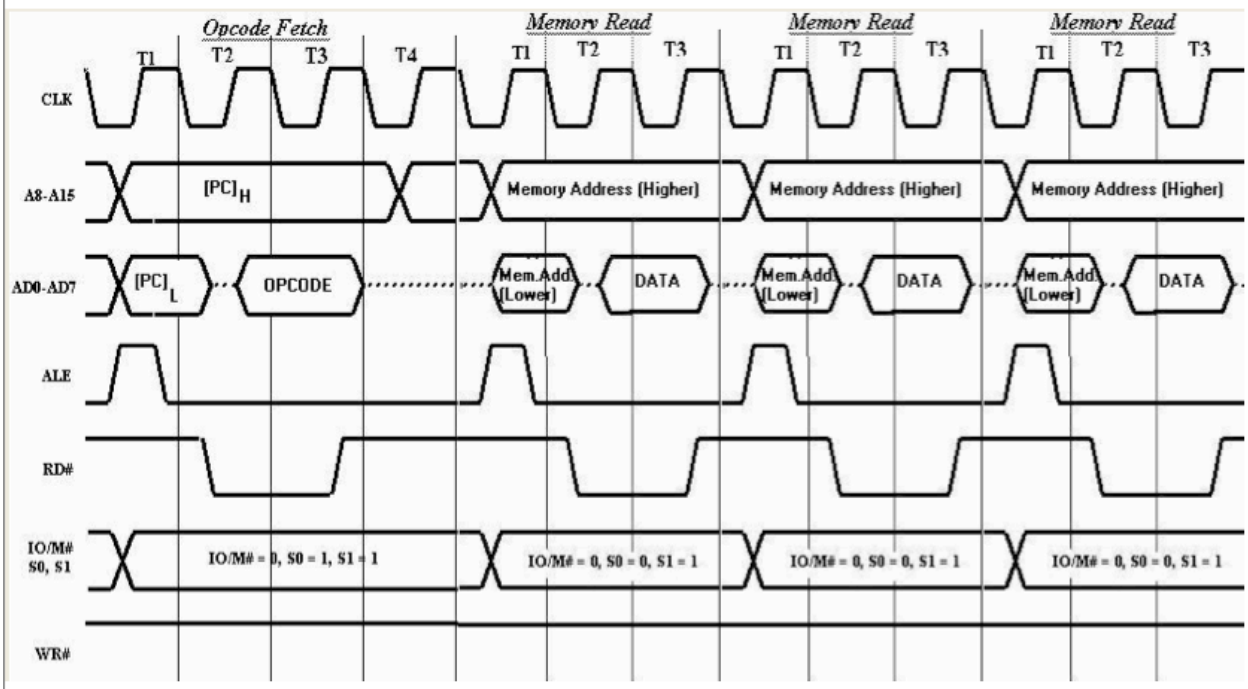


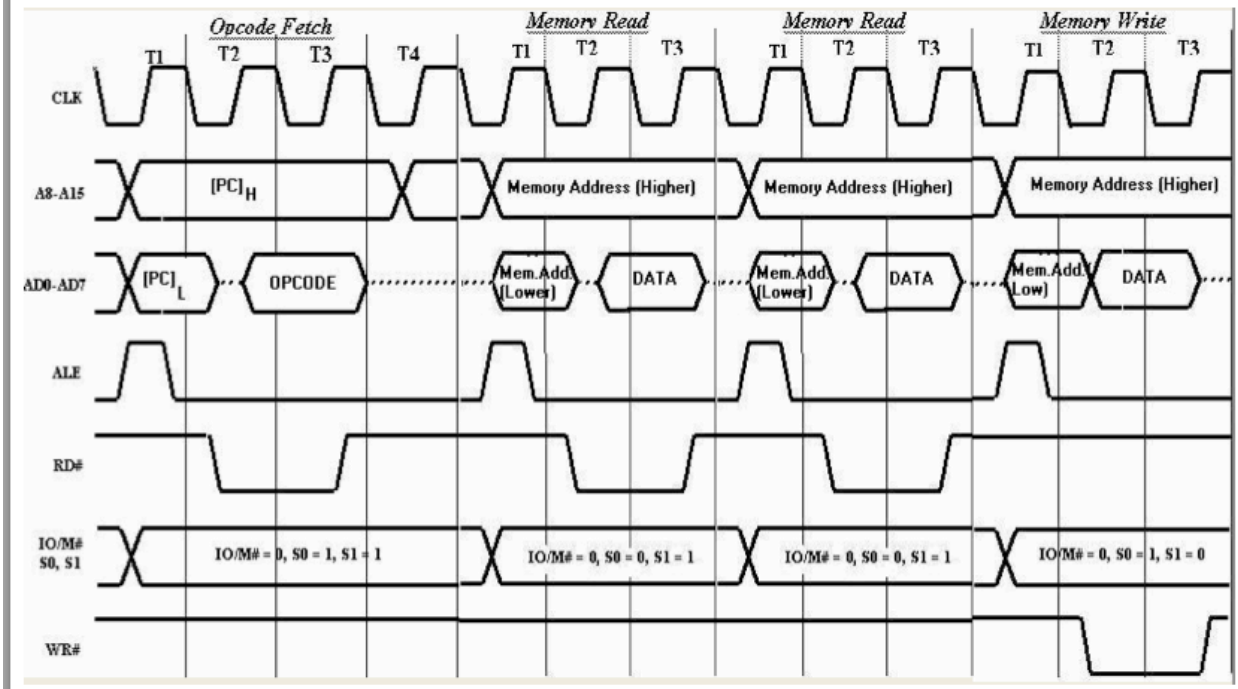
Fig: Timing diagram of I/O read cycle

Other examples:

LDA 16bit Addr



STA 16bit Addr



APPENDIX: RTL

Instruction	MVI A, data
RTL (execute cycle)	t1: MAR ← PC
	t2: MBR ← (MAR) , PC++
	t3: register A ← MBR
Instruction	MVI M, data
RTL	t1: MAR ← PC
	t2: MBR ← (MAR), PC++
	t3: MAR ← registers H&L
	t4: (MAR) ← MBR
Instruction	LXI B, data
RTL	t1: MAR ← PC
	t2: MBR ← (MAR), PC++
	t3: register C ← MBR
	t4: MAR ← PC
	t5: MBR ← (MAR), PC++
	t6: register B ← MBR
Instructions	LDA, address
RTL	t1: MAR ← PC
	t2: MBR ← (MAR++ MAR)
	t3: MAR ← MBR
	t4: MBR ← (MAR)
	t5: register A ← MBR
Instruction	LDAX B
RTL	t1: MBR ← register B & register C
	t2: MAR ← MBR
	t3: MBR ← (MAR)
	t4: register A ← MBR
Instruction	MOV A, B
RTL	t1: MAR ← (Address of B)
	t2: MBR ← (B)
	t3: MAR ← (Address of A)
	t4: A ← MBR