* Register Transfer Language.

The operations executed on data stored in registers are called micro-operations & the result may be stored in the same or another register.

Ex: Shift, Count, Clear & load, MUL, ADD, Move

The sequence of microoperations in a computer can be explained in words, but it will certainly be a lengthy process.

So,
the more convenient & concise way is a symbolic representation to describe the sequence of transfers between registers & the various arithmetic & logic micro-operations.

The symbolic notation used to describe the microoperation transfers among registers is called a register transfer language.

Registers :

Registers are a type of computer memory used to quickly accept, store & transfer data & instructions that are used immediately by the CPU, such CPU's are called Processor registers.

A processor register may hold an instruction, a storage address, or any data (bit sequence or individual characters).

Register is a group of flip-flops capable of storing n - bit information.

Types of Registers:

1. MAR: Register that holds an address for the memory unit is known as memory address register.

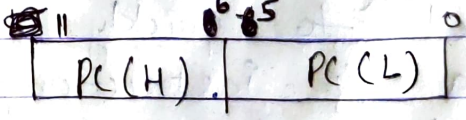
2. DR: Data registers.
It is a 16-bits memory register that is used to hold memory operand.

3. AR: Address registers
It is 12-bits register, used to hold addre for memory.

4. AC: Accumulator.
It is a 16-bits processor register. It is used to store data taken out from the memo

5. IR: Instruction registers.
12-bits register used to hold instruction

6. PC: Program counter.
It is 12-bits register used to hold address of instruction.



7. TR: Temporary register.
12-bits register used to hold temporary regi

8. INPR: Input register (8-bit)
Holds your input data.

9. OUTR: Output register (8-bit)
Holds output data.

L: Lower byte.
H: Higher byte.

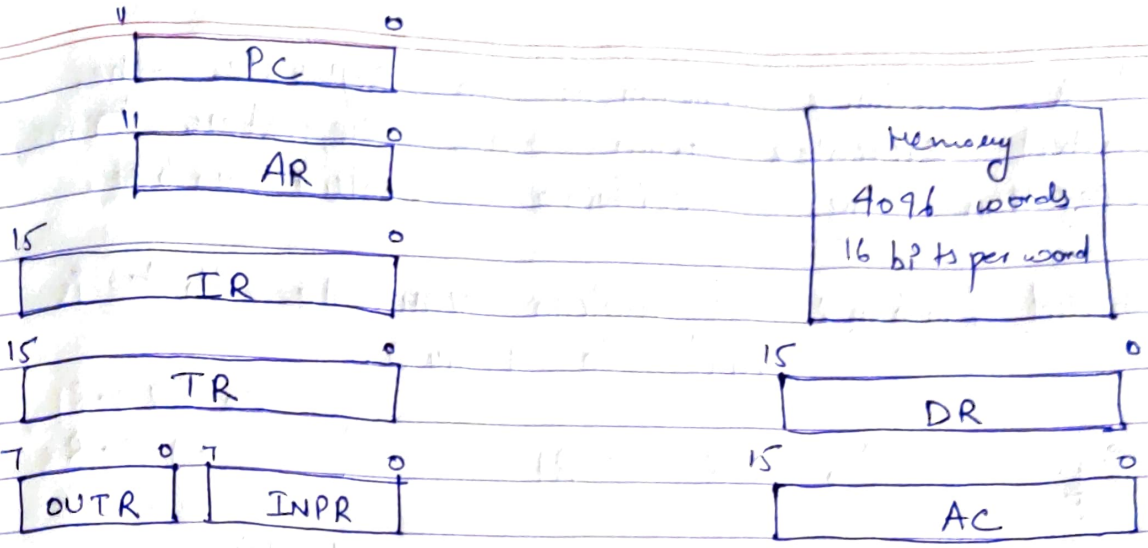


Fig: Basic computer registers & memory.

Register Transfer:

Symbol	Description	Example.
Letters (Number)	Denotes a register	MAR, R2.
Parentheses ( )	Denotes a part of register	R2(0-7), R2(L)
Arrow ←	Denotes transfer of information	R2 ← R1
Comma ,	Separates two microoperations	R2 ← R1, R1 ← R2

Explanations:

1. R2 ← R1
   Replacement of content of R2 with the content of R1. The content of the source register R1 does not change after the transfer.

2. If (P == 1) then (R2 ← R1)
   P: Control signal.

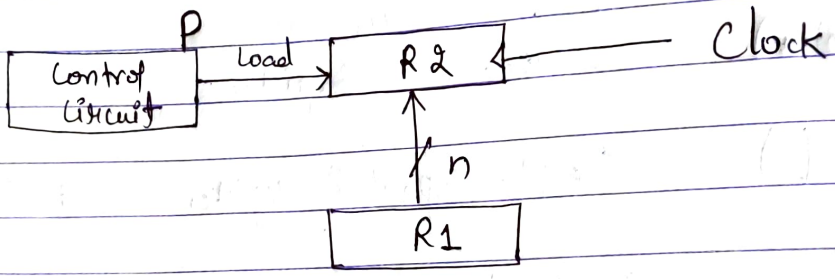
It is sometimes convenient to separate the control variables from the register transfer operation by specifying control function.

Control function: A boolean variable that is equal to 1 or 0.

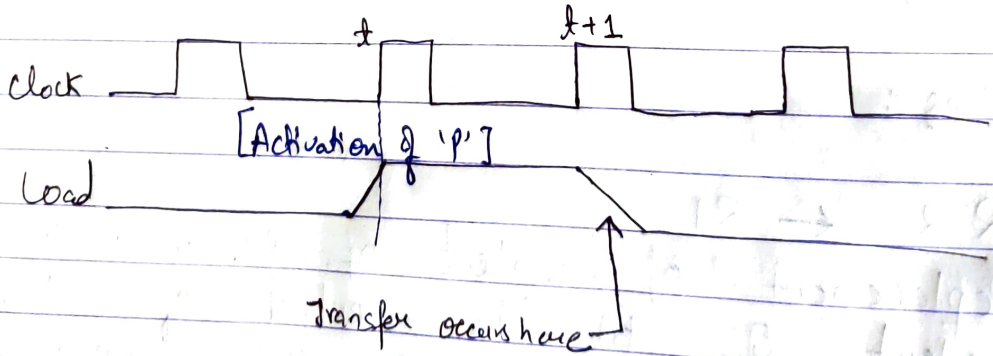
Eg: P : R2 ← R1

{ T : R2 ← R1, R1 ← R2 }

⇒ The control condition is terminated with a colon.
⇒ It represents that the transfer operation should be executed by the hardware only if P=1.



a) Block diagram



b) Timing diagram

Fig: Transfer from R1 to R2, when P = 1

Explanation:

The 'n' outputs of register R1 are connected to the 'n' inputs of register R2.

n : any number of bits for the register.

Register R2 has a load input that is activated by the control variable 'P'.

It is assumed that the control variable is synchronized with the same clock as the one applied to the register.

'P' is activated in the control section by the rising edge of a clock pulse at time 't'.

The next positive transition of the clock at time 't+1', finds the load input active & the data inputs of R2 & then loaded into the register in parallel.

'P' may go back to '0' at time 't+1' otherwise the transfer will occur with every clock pulse transition while 'P' remains active.

The transfers occur during a clock edge transition.