using a stack for the return address is that when a succession of subroutines is called, the sequential return addresses can be pushed into the stack. The return from subroutine instruction causes the stack to pop and the contents of the top of the stack are transferred to the program counter.

- A subroutine call is implemented with the following microoperations:

SP ←SP - 1	Decrement stack pointer
M[SP] ←PC	Push content of PC onto the stack
PC ← effective address	Transfer control to the subroutine

- If another subroutine is called by the current subroutine, the new return address is pushed into the stack, and so on. The instruction that returns from the last subroutine is implemented by the microoperations:

PC←M[SP]	Pop stack and transfer to PC
SP ←SP + 1	Increment stack pointer

Control Unit:

The fundamental concepts forming the basis for control unit design are the register transfer operations and their analytical descriptions. A digital processing system is a collection of registers where a processing activity is performed by a sequence of data-transfer operations among the registers either directly or with processing hardware (ALU). Consider the simple transfer shown in Figure 4.1

Here, 8 bits of information are to be moved from register A to register B. This means transferring a copy of A to B.

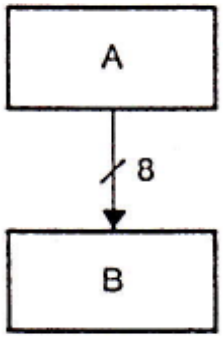

Figure: A simple register transfer from A to B.

Generally two inputs are associated with each register:

- Enable input or control input
- Data input

The enable input controls the data flow from register A to B. The B register of Figure 4.3 is loaded with the contents of the A register only when the enable input E is held high; otherwise the contents of the register remain the same. Such a conditional transfer can be expressed as

E: B ← A

A possible hardware implementation of a register with an enable input is shown in Figure 4.4.

The bit B_i in this figure is only loaded with bit A_i when the enable input E is high. The purpose of the enable input is to make sure that data transfer between the A and B registers takes place only under a predetermined condition and not after every clock pulse. For this reason, this input is called the *control input* and is driven by the control unit.

Sometimes, the control input may be a function of more than one variable. Consider the following register transfer involving three 8-bit registers A, B, and D:

IF A > B and D [0] = 0 then A ← B

The condition A > B can be determined by using an 8-bit comparator. If we assume that the comparator output G goes high when A > B, then this conditional transfer can be described as follows:

C_0: A ← B where $C_0 = G \wedge D[0]'$

The hardware setup corresponding to this situation is shown in Figure 4.5.

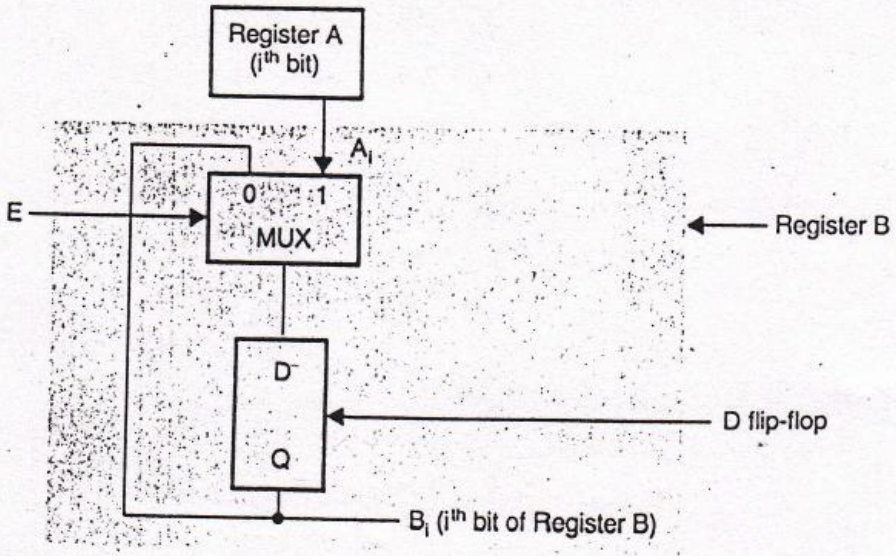

Figure: Hardware implementation of a Register with Enable input.

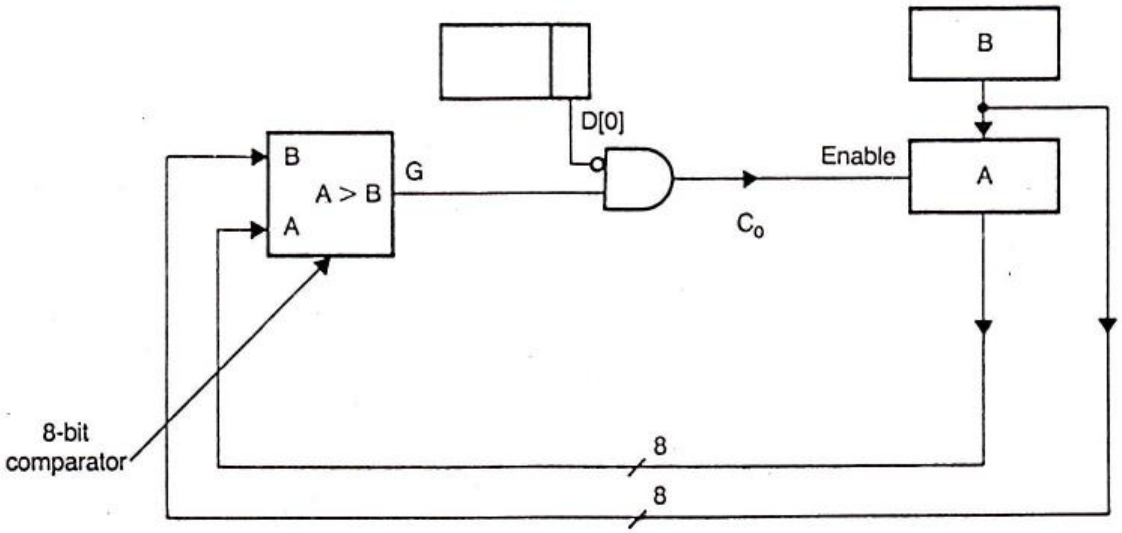

Figure 4.5 Hardware Implementation of C_0: A ← B where C_0 = G D [0]'

It may often be necessary to perform some register transfer operation that involves selection. For example:

if x = 0 and t = 1 then A ← B
else A ← D

Where A, B, and D are 8-bit registers, such a selective register transfer can be expressed as follows:

$C_0 : A \leftarrow B$

$C_0' : A \leftarrow D$

where $C_0 = x't$ and $C_0' = (x't)' = x + t'$.

A hardware implementation for the preceding situation is shown in Figure 4.6.

The MUX of Figure 4.6 selects register B if $C_0 = 1$; otherwise register D is selected.

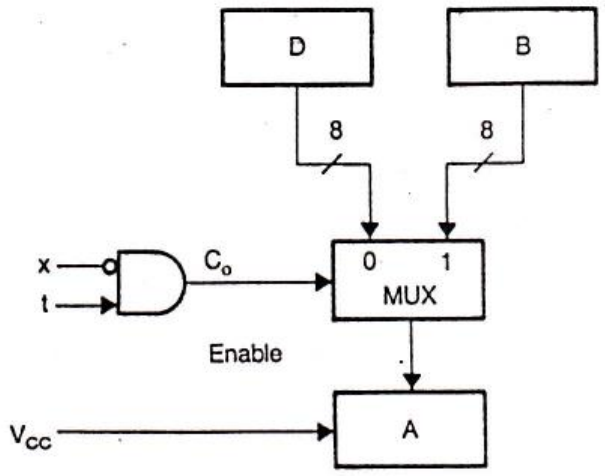

Figure 4.6 Hardware Implementation if x = 0 and t = 1, then A ← B else A ← D