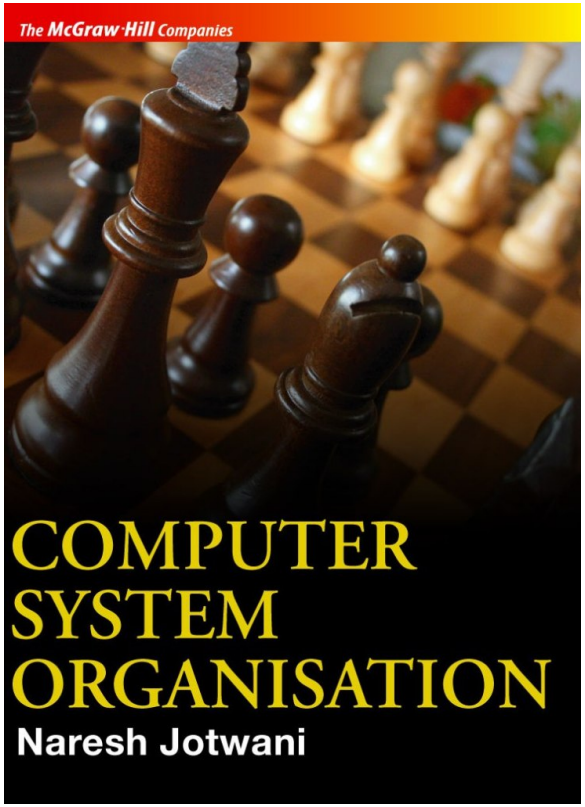
Mc Graw Hill Education

COMPUTER SYSTEM ORGANISATION

Naresh Jotwani

PowerPoint Slides



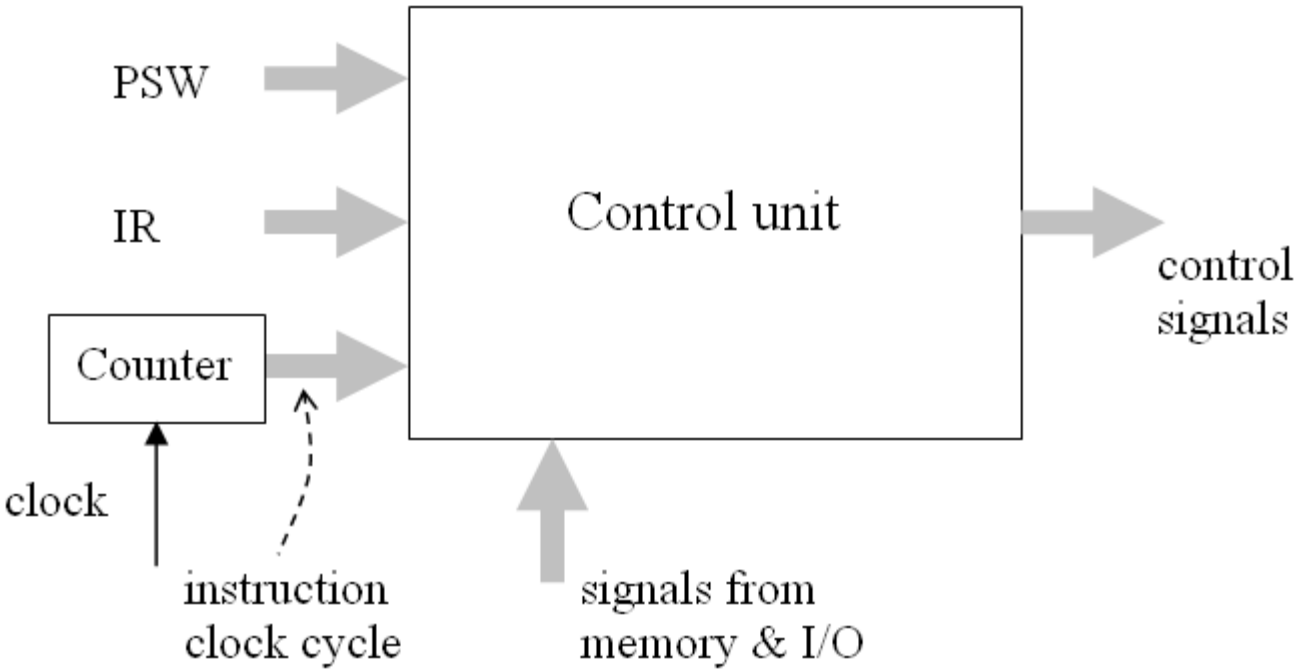
CHAPTER 7

CONTROL UNIT

Introduction:

- Control unit of the processor generates the control signals to activate the required micro-operations in each clock cycle.
- A typical hardwired control unit is a combinational circuit, with the inputs and outputs as shown in the next slide.
- The combinational circuit realizes the required functional relationship between the input signals and the output signals of the control unit.

- A crucial input to the control unit is supplied by the counter – shown on the lower left in the figure – which tracks the instruction clock cycle.



Implementing a typical Instruction:

- Let us examine the micro-operations and control signals required for the machine instruction below:

ADD R3, R4

- We shall see:

  (a) how the fetch and execute phases of this machine instruction are broken down into micro-operations, and

  (b) what control signals are required to carry out these micro-operations.

Micro-operations & control signals required:

Instruction clock cycle	Micro-operations required	Control signals required
0	Initiate memory read for instruction	PC_{out}, ADR_{in}, READ
1	Move instruction into IR	$DATA_{out\text{-}B}$, IR_{in}
2	Move register operands to ALU, increment PC	$R3_{out\text{-}A}$, $AIN\text{-}1_{in}$, $R4_{out\text{-}B}$, $AIN\text{-}2_{in}$, PC_{inc}
3	Perform ALU operation	ALU_{ADD}
4	Move result to R3, reset control unit counter	$AOUT_{out}$, $R4_{in}$, CTR_{reset}

Operations to be carried out in each of the five instruction clock cycles of the instruction:

CC 0: Contents of PC moved to address register, and READ control signal is generated for cache memory.

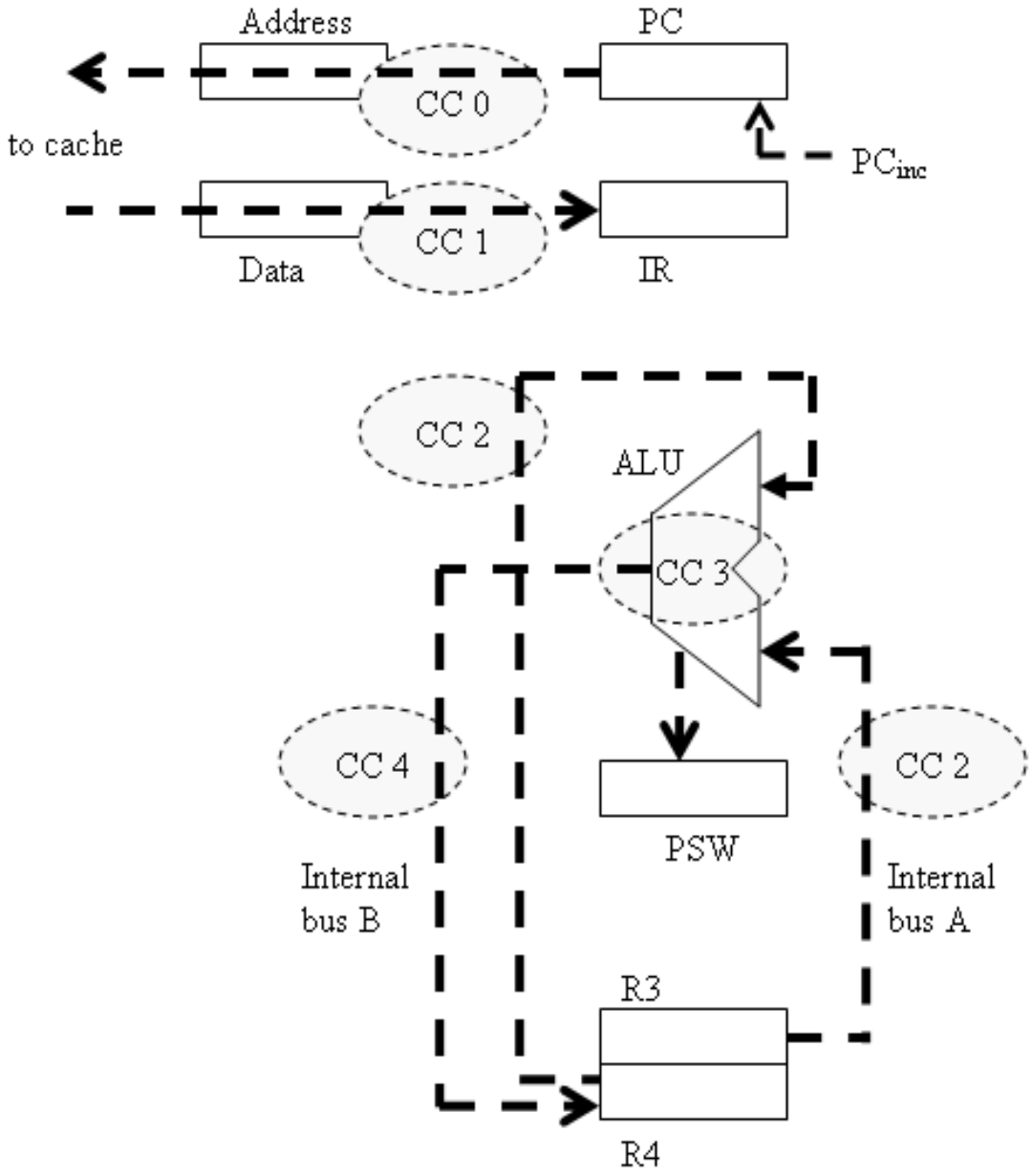
CC 1: Word read from cache is moved into IR, for instruction decoding.

CC 2: Contents of R3 and R4 are moved into the two ALU inputs IN-1 and IN-2 respectively. PC is incremented.

CC 3: Addition operation is performed in ALU.

CC 4: Contents of ALU output register are moved into R4. Instruction cycle counter of the control unit is reset to zero for the next instruction.

Diagram showing operations in the five clock cycles of instruction:



- Labels from ‘CC 0’ to ‘CC 4’ in the diagram above represent operations taking place in the five clock cycles of the instruction.

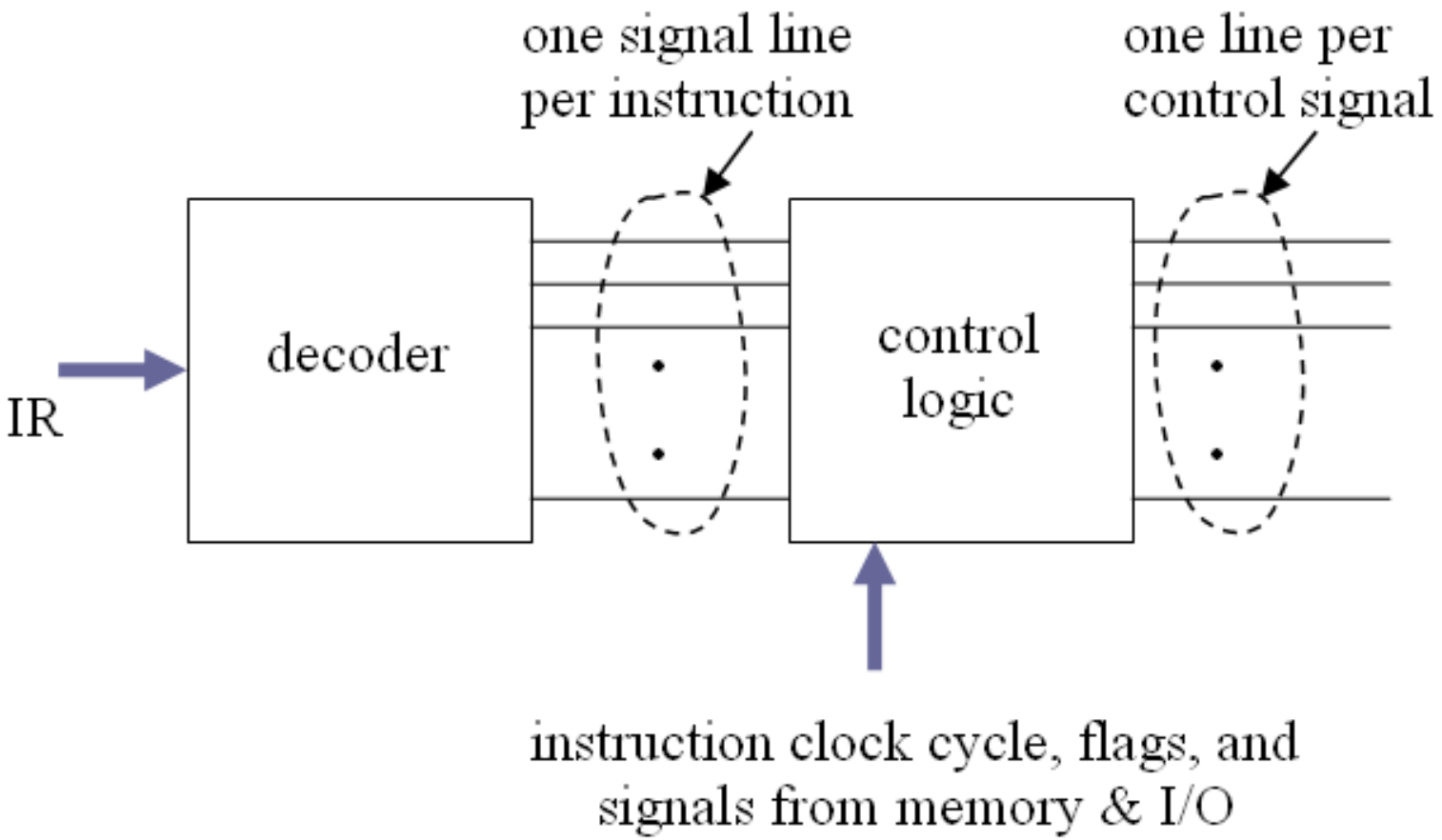
- Clock cycles ‘CC 0’ and ‘CC 1’ make up instruction fetch and decode stages IF and ID (ref. Chapter 6).

- The remaining three clock cycles CC 2, CC 3 and CC 4 make up the OP, MA and RS stages (ref. Chapter 6).

Hardwired control unit:

- It is a combinational circuit which realizes the required functional relationship between its inputs and outputs
- Basic organization of hardwired control unit is shown in the next slide.
- The first part of the control unit, shown on the left in the figure, is the *instruction decoder.*

Instruction fetched into IR must be decoded, so that the appropriate control signals for it can be generated in each instruction clock cycle.



- Output of the instruction decoder has – in theory – one separate signal line for each instruction, addressing mode, and instruction modifier.

- These lines and other inputs – i.e. instruction clock cycle, flags, and signals from memory & I/O – are inputs to the control logic, which produces the required control signals.

- This control logic is shown schematically on the right in the figure.

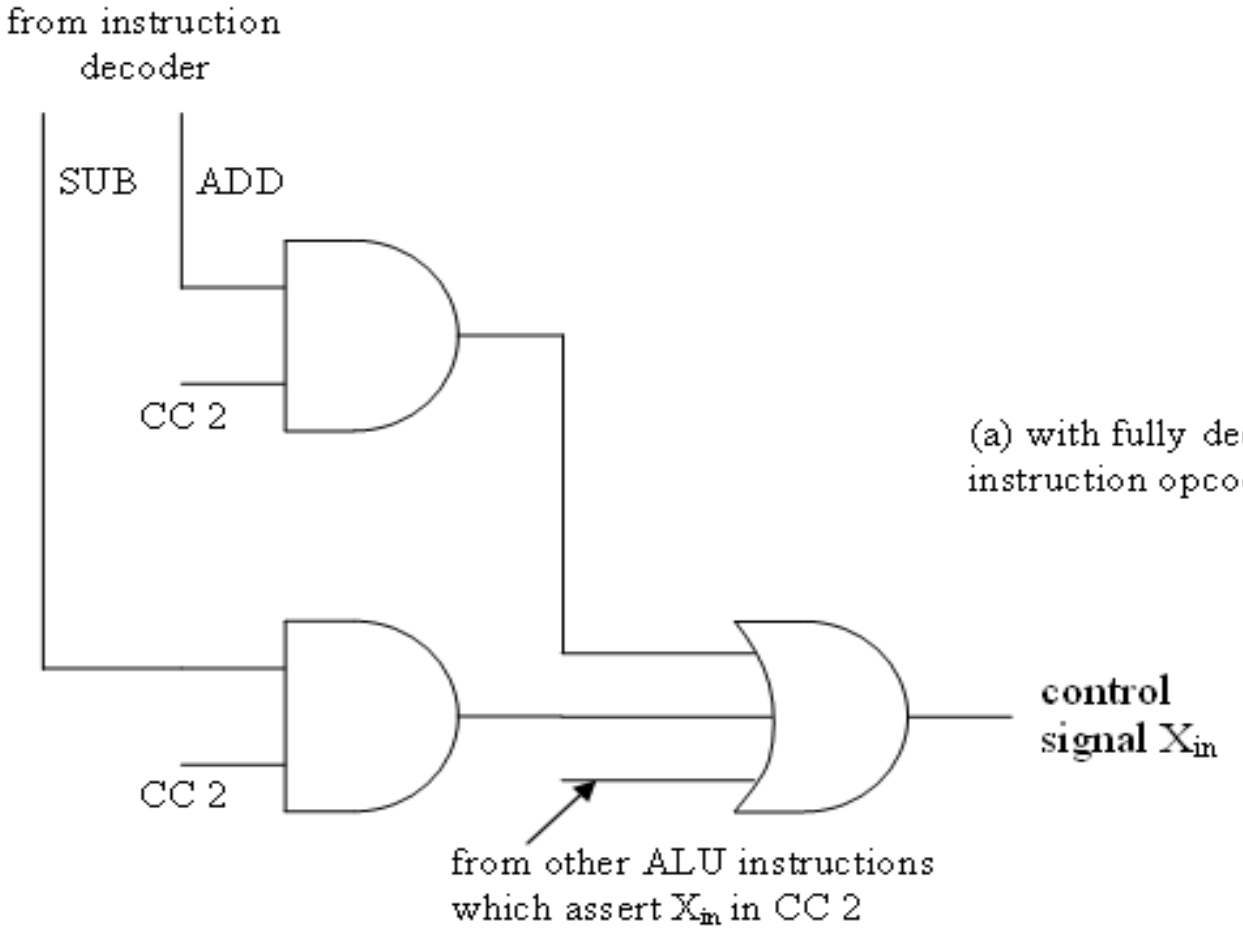
- Suppose processor designers need that a control signal X_{in} must be generated in clock cycle CC 2 of ADD, SUB, and some similar instructions involving ALU operations.

- Control logic in part (a) of the next slide satisfies this design requirement.

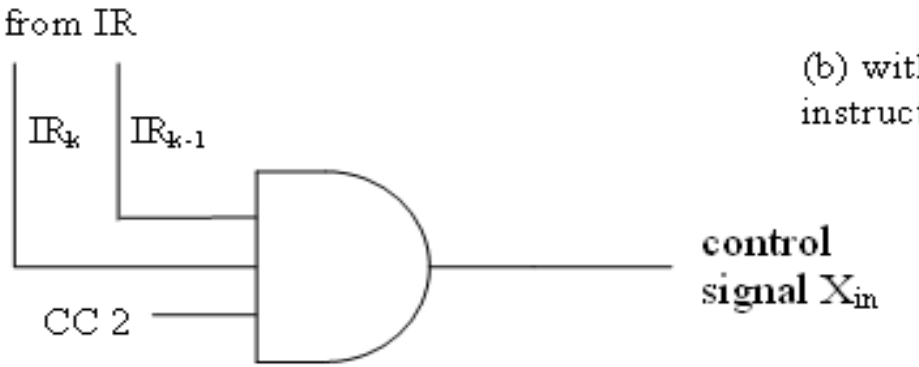
  - Output of the first *and* gate is enabled for instruction ADD in CC 2.
  - Output of the second *and* gate is enabled for instruction SUB in CC 2, and so on for other instructions.

- Thus there is one *and* gate in this part of the circuit for each combination of instruction and clock cycle which must generate the signal X_{in}.

- The outputs of all these *and* gates are sent as inputs into the *or* gate, so that the control signal X_{in} is enabled for all the required combinations of instructions and clock cycles.



(a) with fully decoded instruction opcodes



(b) with direct inputs from instruction register (IR)

- Part (b) of the figure shows a simpler solution to generate the required control signals using binary encoding of instructions.

- Assume the following property of binary instruction codes:

  For all instructions requiring control signal X_{in} in clock cycle CC 2 – and only for these instructions – the two bits IR_k and IR_{k-1} in the instruction are set to '1'.

- Now note that, whenever bits IR_k and IR_{k-1} of the instruction are '1', the *and* gate output is '1' in clock cycle CC 2.

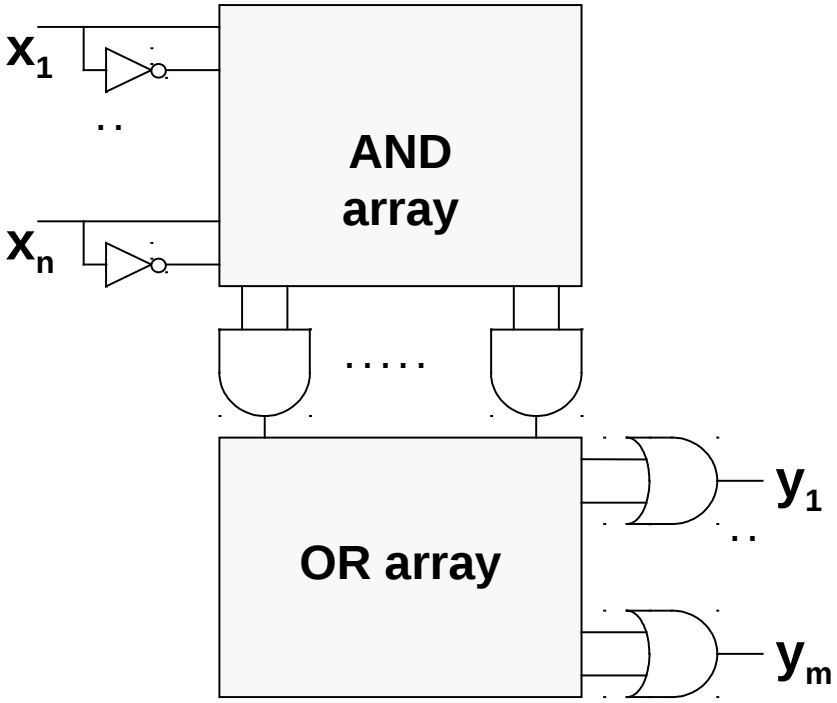
- The single *and* gate takes care of ALL instructions which require X_{in} to be enabled in CC 2, and so a separate decoded line for each of these instruction is not needed

- *Programmable logic devices* can be used for hardwired control units.

- A *programmable logic array* (PLA) is a typical example.
  - A highly regular structure
  - Logic is 'programmed' into it in the form of connections made between row lines and column lines of two arrays.
  - Circuit components are made small, and operate at low voltages, giving high speeds and low power consumption.

- Sophisticated design and verification tools result in faster design times and lower costs.

- A typical PLA with *n* inputs and *m* outputs is shown in schematic form in the next slide.



- As seen on the left: PLA allows any input signal x_i to be used directly, or in inverted form $\neg x_i$.
- AND array allows sets of inputs – either direct or inverted – to be combined using *and* gates.
- Outputs of AND array are fed to the OR array, where they can be combined using *or* gates.

A limitation of hardwired control:

- Consider the following *block copy* instruction

```
BCOPY  Ri, Rj, Rk
```

- The three register operands specify, respectively :
  - a *source address* in main memory
  - a *destination address* in main memory
  - a byte *count*.

- The specified number of bytes must be copied from memory locations starting at the source address, to locations starting at the destination address.

- BCOPY is a hardware implementation of the copy function seen earlier in software. Note that the byte count may vary from zero to tens of thousands.

- Let *C* denote the value of this count as specified by the programmer.
  - In the execute phase of this instruction, we need to copy *C* bytes from one location to another location in main memory.
  - Thus an iterative action must be performed *C* times in the execute phase of this one instruction.
  - Therefore the number of processor clock cycles required in the execute phase of this instruction will be *b* x *C*, for some constant *b* (i.e. *b* cycles per iteration).

- With a hardwired control unit, a machine instruction takes a fixed number of clock cycles *n*, which does not depend on the operands specified by the programmer.

- But, for BCOPY instruction, the number of clock cycles required in execute phase is *b* x *C* – and clearly it does depend on the count *C* specified by the programmer.

- Therefore a hardwired control unit of the type seen earlier will not suffice for machine instructions such as BCOPY.

- In general, if a processor has complex instructions in its instruction set, then a combinational circuit cannot provide all the functions of the control unit.

- For complex instructions, an alternative technique for designing the control unit had been developed since relatively early days of computer technology.

- Microprogrammed control

  – This alternative technique yields slower control units, but it can accommodate instructions of virtually any complexity.

Microprogrammed control:

Basic concepts:

- C*ontrol word* - stored at each location in *control memory*. Bits in control word specify control signals to be enabled in a particular instruction clock cycle.

- *Control memory* (or *control store)* - Memory element in which all the control words are stored.

  As control words are read out from control memory, one after another, they enable control signals of the processor.

- *Control sequence* - A sequence of control words which are read out from the control memory, one after another.

- Consider the two input, two output combinational circuit shown in part (a) of the next figure. Its inputs are labeled X_1 and X_2, while the outputs are labeled Y_1 and Y_2.
- In part (b) of the figure, one possible truth table of the circuit is shown. For each possible combination of values of X_1 and X_2, the table gives the corresponding values of Y_1 and Y_2.
- The truth table can be stored in a memory element with four locations, with two bits per location.
- X_1 and X_2 specify the memory address.
- Y_1 and Y_2 are the two bits stored per location.

A combinational circuit with two inputs and two outputs:

(a) Block diagram

Input X_1	Input X_2	Output Y_1	Output Y_2
0	0	0	1
0	1	1	1
1	0	1	1
1	1	1	0

(b) Truth table

- The memory element has the same inputs and outputs as the circuit block of part (a) of the figure, except that it also has a *read* control signal coming in to it.
- Thus the memory element can provide an alternative to logic gates for the combinational logic needed.
- This method is known as *table look-up*.
- For a combinational circuit with m inputs and n outputs, the memory element would need 2^m locations, where each location stores n bits.
- But, as noted earlier, memory elements offer slower speeds of operation than logic gates

- In designing a control unit, the *flexibility* offered by the memory element can sometimes offset the speed advantage of logic gates.
- Useful for complex instructions, e.g. BCOPY.
- A control unit based on this principle is known as a *microprogrammed* control unit.
- Of course – as we shall soon see – such a control unit is more complex than a simple memory element providing table look-up.

Basic design of microprogrammed control:

(a) Store a single copy of the *fetch* control sequence in control memory, to be used by all the machine instructions.

(b) Store a single control sequence for each machine instruction in control memory

(c) If two or more machine instructions have a common part in their control sequences, then store only a single copy of the shared part of control sequence.

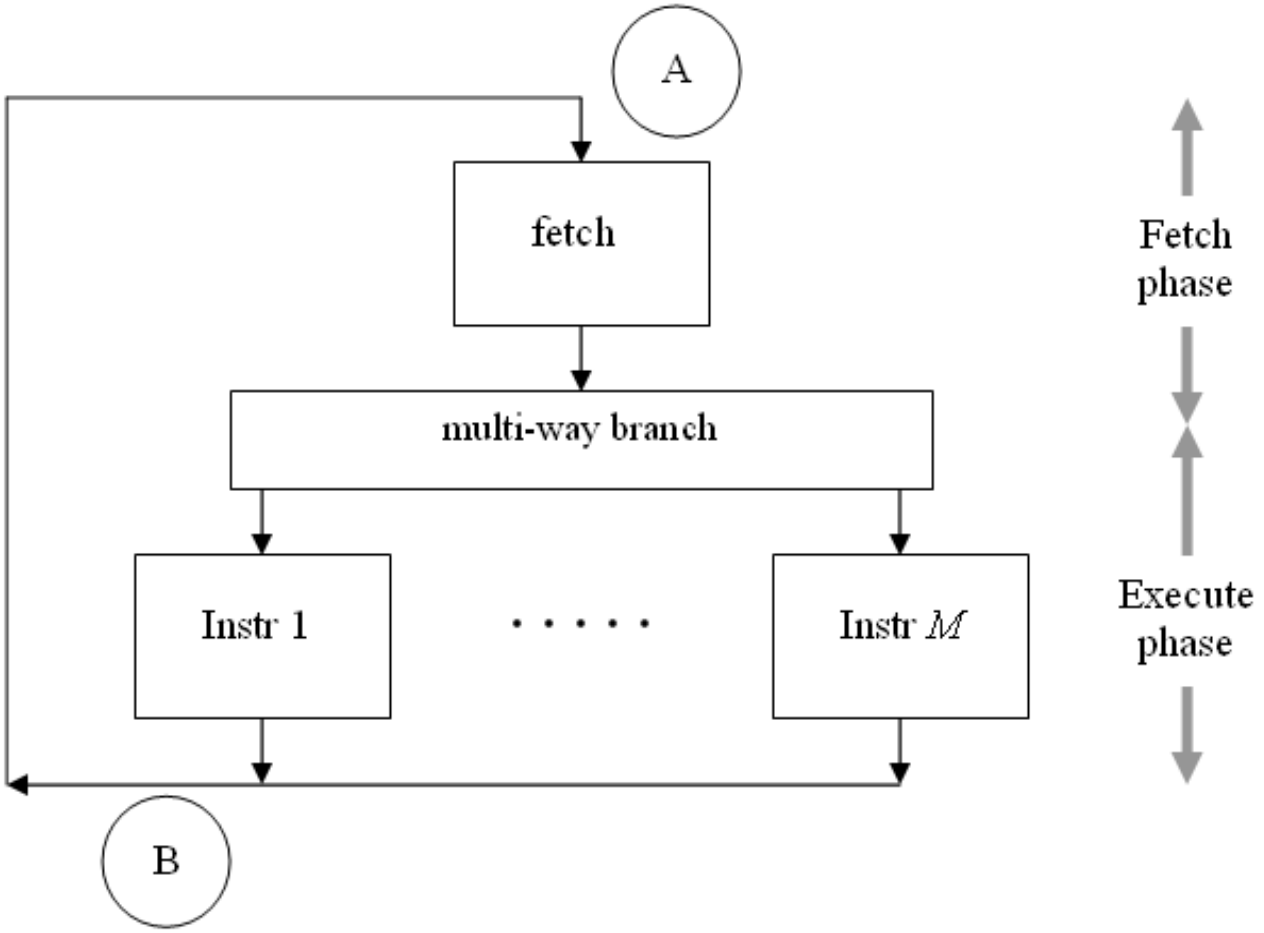
Microinstruction sequencing:

- The next figure depicts a structure of control sequences, with a common *fetch* control sequence for all machine instructions, followed by a separate control sequence for each machine instruction.

- The full set of control sequences in the control store makes up the *control program* – or *microprogram* – of the processor.
  - A single control word in a microprogram is also known as a *microinstruction*.

- The execution of one machine instruction requires the execution of several microinstructions, in as many processor clock cycles.

Broad structure of control sequences in control memory:



- The common *fetch* control sequence (see figure above) results in a machine instruction being fetched into IR.

- Once the machine instruction is in IR, a conditional *jump* takes place in the microprogram to the control sequence of the machine instruction.

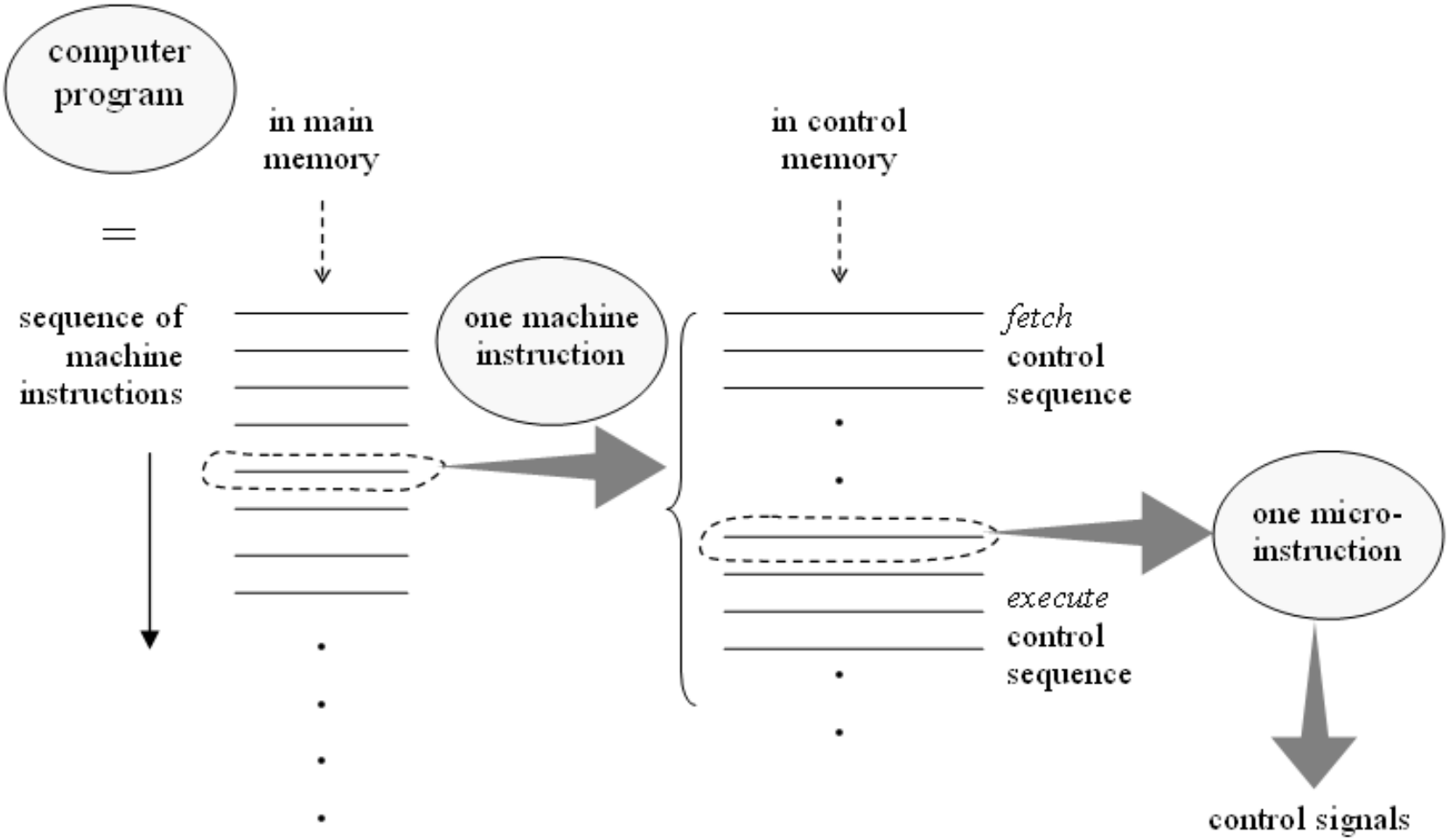
- The conditional *jump* is a 'multi-way branch' – since, on a processor with *M* machine instructions, exactly one of *M* control sequences is selected, depending on the machine instruction in IR.

- The control sequence specific to a machine instruction then causes the processor to go through the *execute* phase of the instruction.

- In the last microinstruction of the *execute* phase, an unconditional 'backward' *jump* takes place to the common *fetch* sequence - for the next machine instruction to be fetched.
- At this time, the instruction clock cycle is also reset to zero.
- Combined effect of a sequence of micro-instructions is the execution of a machine instruction.
- Trace out exactly one of *M* possible paths from the point marked 'A' to the point marked 'B' (see figure above).
- The sequence of microinstructions on this path makes up *fetch* and *execute* phases of one machine instruction.

- The next figure shows the relation between a computer program, machine instructions, and microinstructions.
- A computer program is a sequence of machine instructions.
- The execution of a single machine instruction involves multiple microinstructions, in as many processor clock cycles.
  - *Fetch* control sequence contains microinstructions which are common to all machine instructions
  - *Execute* control sequence varies from one machine instruction to another.
- In each clock cycle, a microinstruction generates the processor control signals required

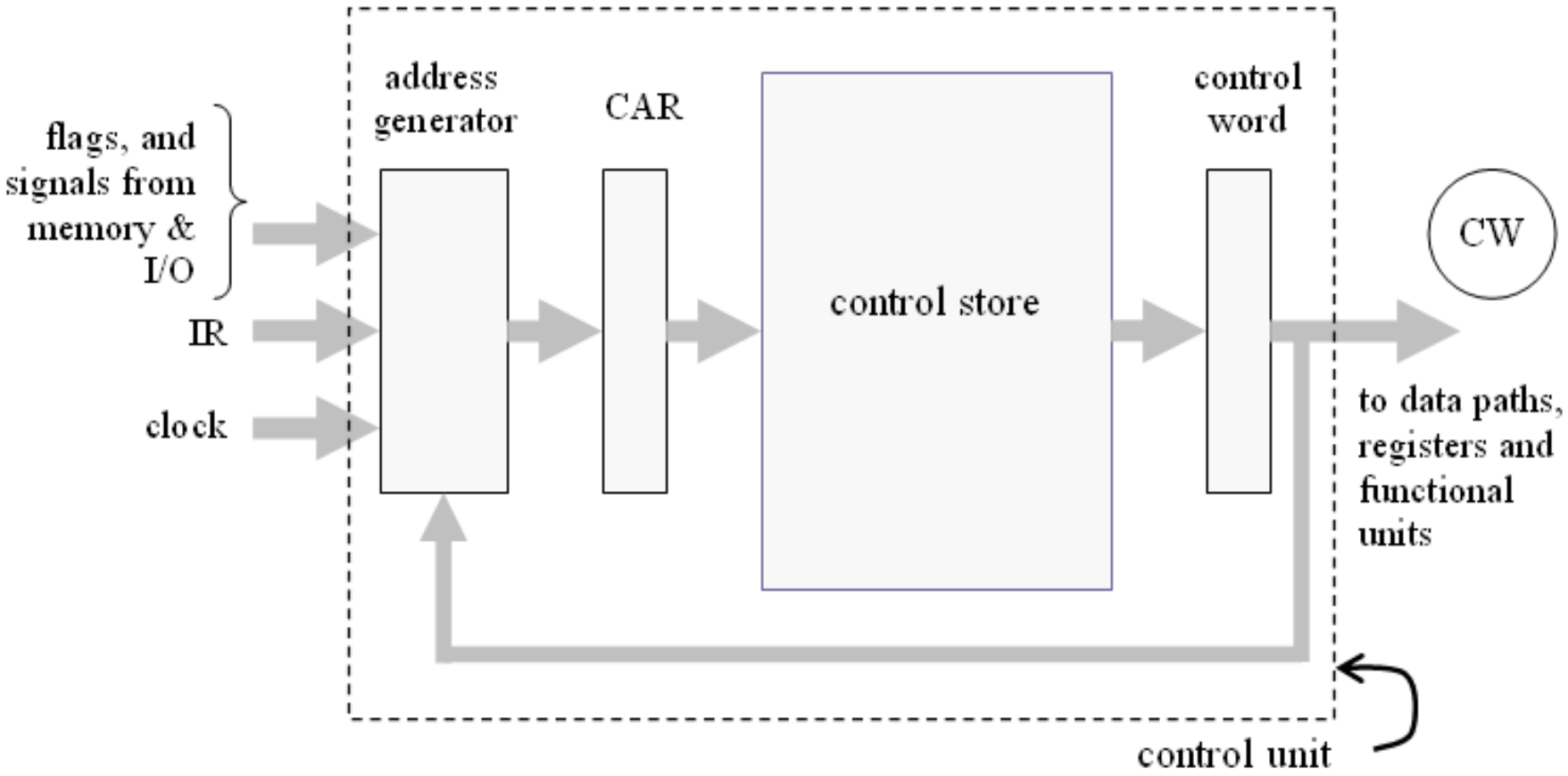
Relation between computer program, machine instruction and micro-instruction:



Microprogrammed control unit:

- Next figure shows the typical internal physical organization of a microprogrammed control unit
- Control store is the central component holding the control program.
- *Control address register* (CAR) to the left holds the address of the current microinstruction
- Clock input on the left ensures that one micro-instruction is read from control store in every clock cycle, from the location specified in CAR.

A microprogrammed control unit:



- Control signals of the microinstruction act upon data paths, registers, and functional units within the processor, shown at point ‘CW’.
- A separate *address generator*, shown to the left, provides the next control address to CAR.
- The outer rectangle in broken lines is the *control unit.*
- Control memory or control store is the central component which holds the entire microprogram (*microcode)* for the instruction set.

Microinstruction formats:

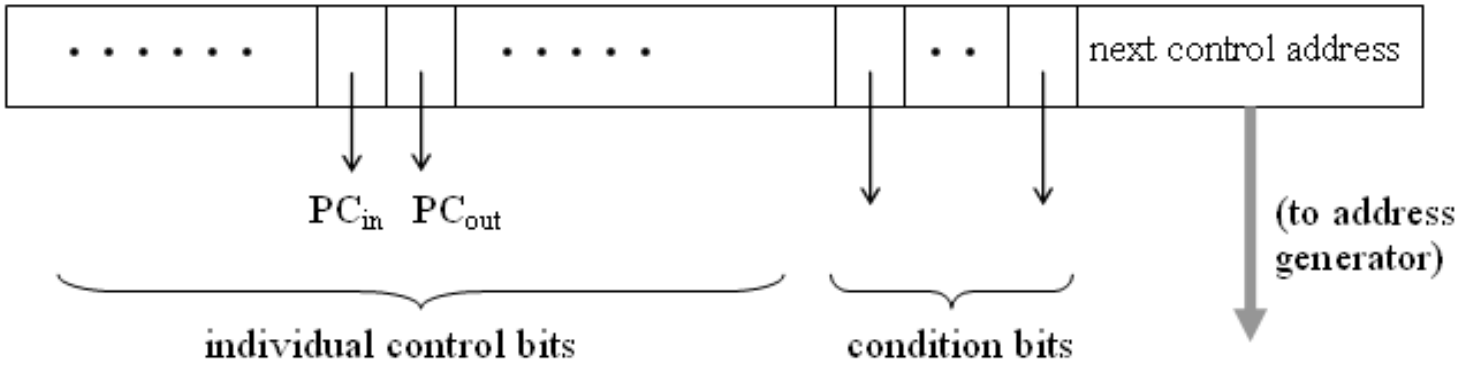
1) Horizontal microinstruction format

- It is the simplest microinstruction format, which provides for individual control bits for control signals, a *next control address* field, and some *condition bits*.

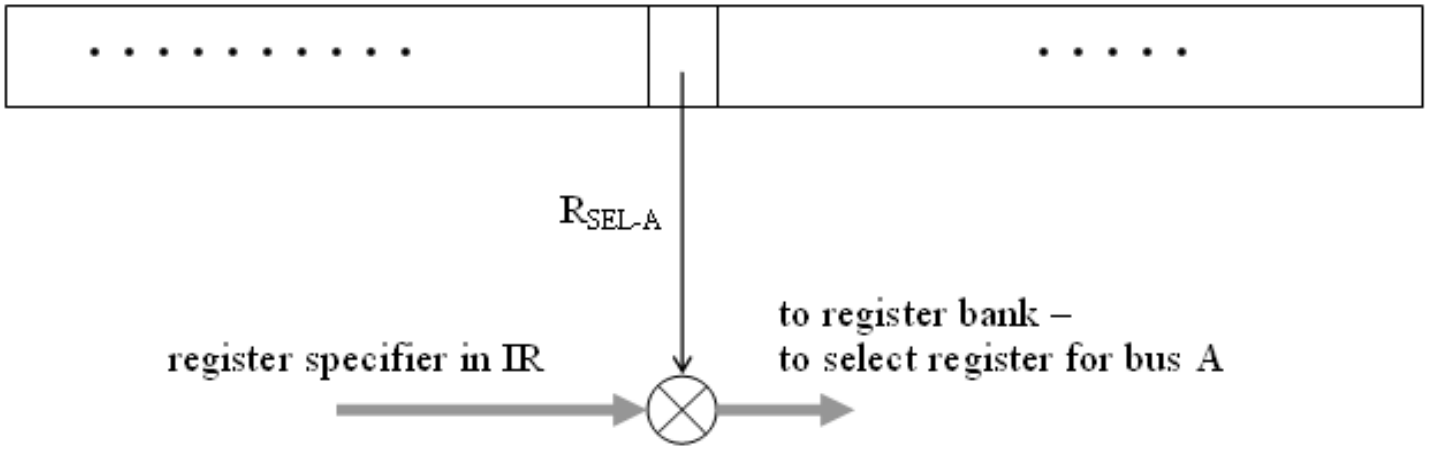
- A typical such format is shown in part (a) of the next figure. In theory, there is one bit in the microinstruction for every control signal.

2) Vertical microinstruction format

- Microinstruction format which uses mostly encoded fields
  - Suppose that there are eight control signals in the processor -- of which it is known that exactly one is active in any processor clock cycle.
  - Then the required control signal can be *encoded* in a field of three bits, reducing the required field width from 8 bits to 3 bits.
  - Fields must be decoded to generate control signals.



(a) Horizontal format



(b) Selection of IR field

3) Direct use of IR bits:

In this arrangement, register-select bits of IR will be decoded in the register bank, and ALU function- select bits of IR will be decoded in the ALU.

In part (b) of previous figure, R_{SEL-A} bit of IR causes the 'register specifier' bits of IR to be gated to the register bank.

Note that this is neither 'horizontal' nor 'vertical' structure!

Comparison with Hardwired Control:

- Hardwired control offers high speed of operation and a more compact implementation -- suitable for a pipelined, high-performance processor.
- Microprogrammed control offers flexibility and extensibility; it can support complex instructions.
- Slower speed of operation and unsuitability for pipelined processors are two major drawbacks of micro-programmed control.
- Some modern processors make use of hardwired and microprogrammed control.

- Choice between these two techniques is governed by:
  - (a) the relative complexity of instruction set
  - (b) the relative importance of speed versus flexibility and extensibility in the design objectives of the processor.

Summary

- Function & design of the control unit; implementation of a typical machine instruction.
- Hardwired control unit; use of a programmable logic device.
- Microprogrammed control unit: Control store, microinstruction sequencing, microinstruction formats.
- Comparison between two types of control units.