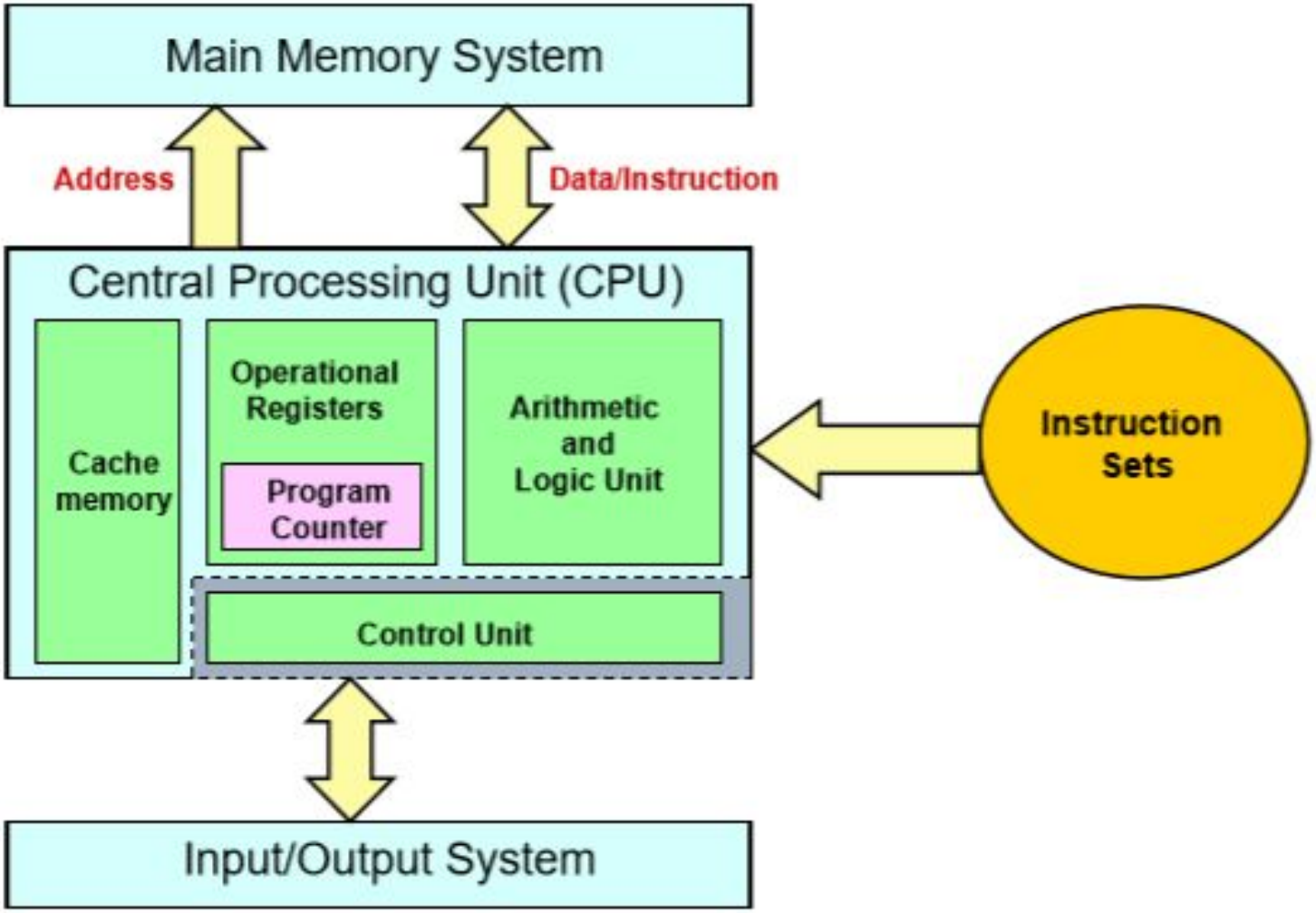
UNIT-5

Basic Processing Unit

Outline

- Fundamental Concepts
- Hardwired Control
- Microprogrammed Control

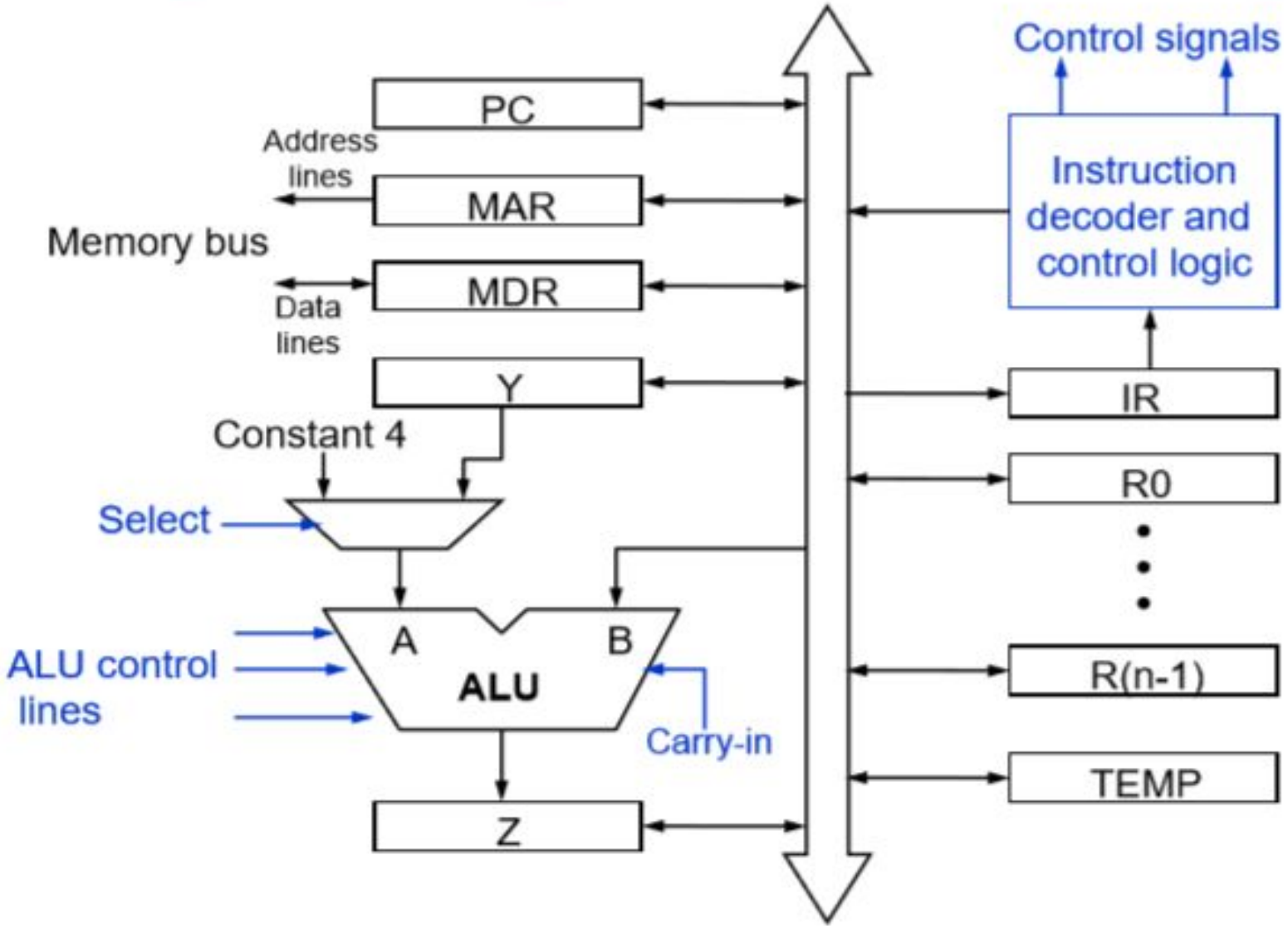
Content Coverage



Instruction Execution

- To execute an instruction, the processor has to perform the following three steps:
  - Step 1: fetch the contents of the memory location pointed by the PC. The contents of this location are interpreted as an instruction to be executed. Hence, they are loaded into the IR. Symbolically, this can be written as IR←[[PC]]
  - Step 2: assuming that the memory is byte addressable, increment the contents of the PC by 4, that is, PC←[PC]+4
  - Step 3: carry out the actions specified by the instruction in the IR
- Step 1 and Step 2→*fetch phase*
- Step 3→*decode phase, execution phase, and/or write phase*

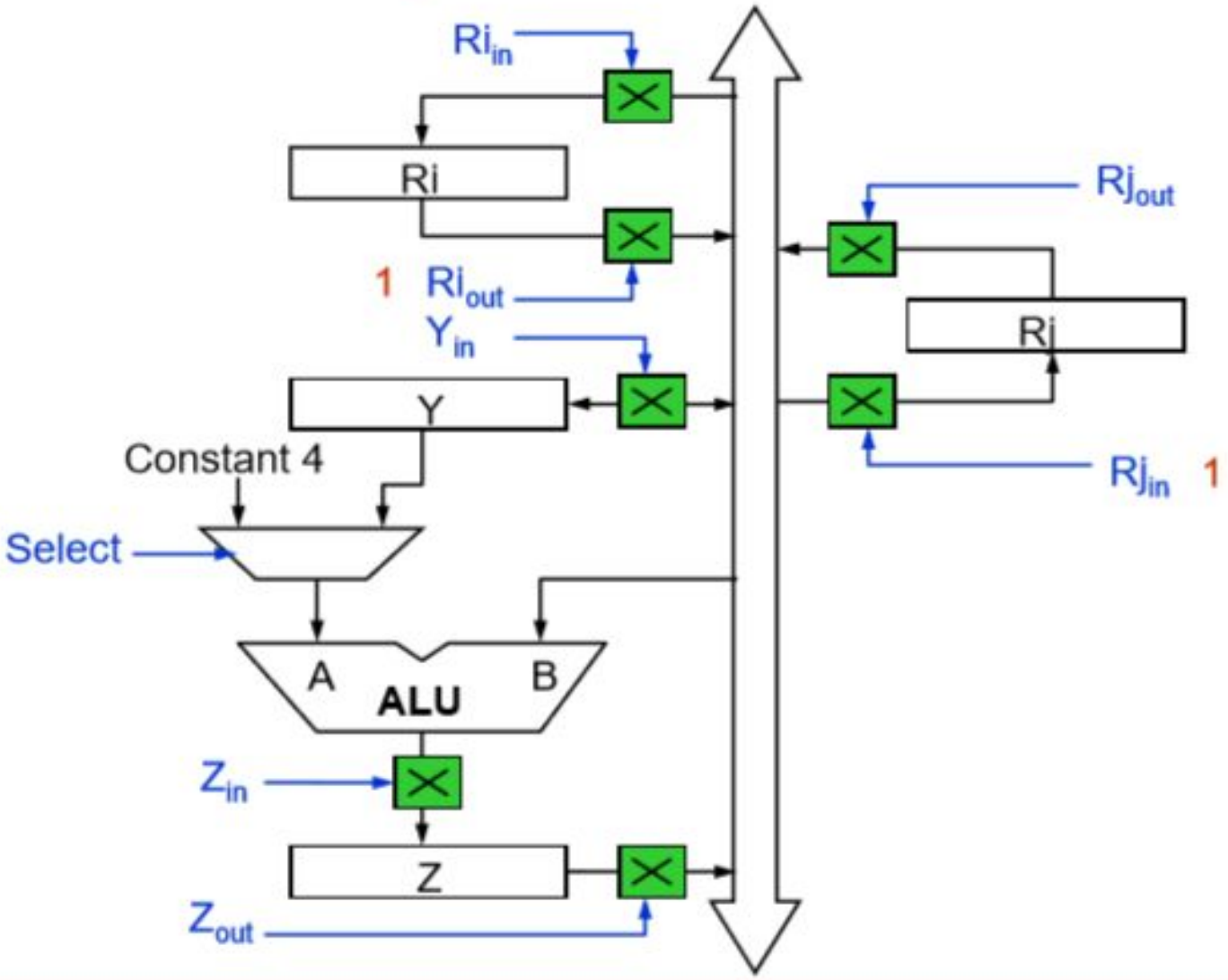
Single-Bus Organization of the Datapath



Instruction Execution

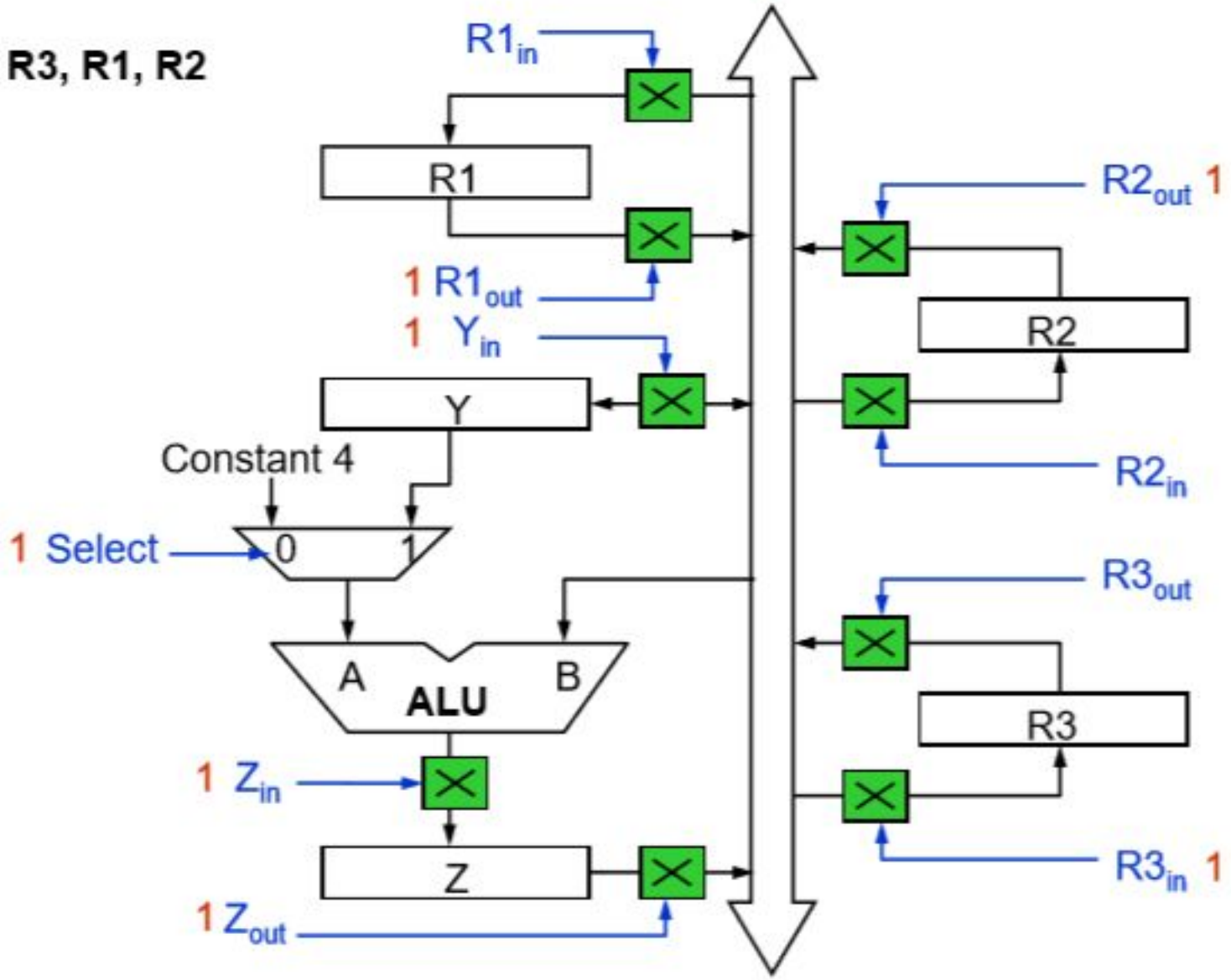
- An instruction can be executed by performing one or more of the following operations in some specified sequence
  - Transfer a word of data from one processor register to another or to the ALU
  - Perform an arithmetic or a logic operation and store the result in a processor register
  - Fetch the contents of a given memory location and load them into a processor register
  - Store a word of data from a processor register into a given memory location

Register Transfers

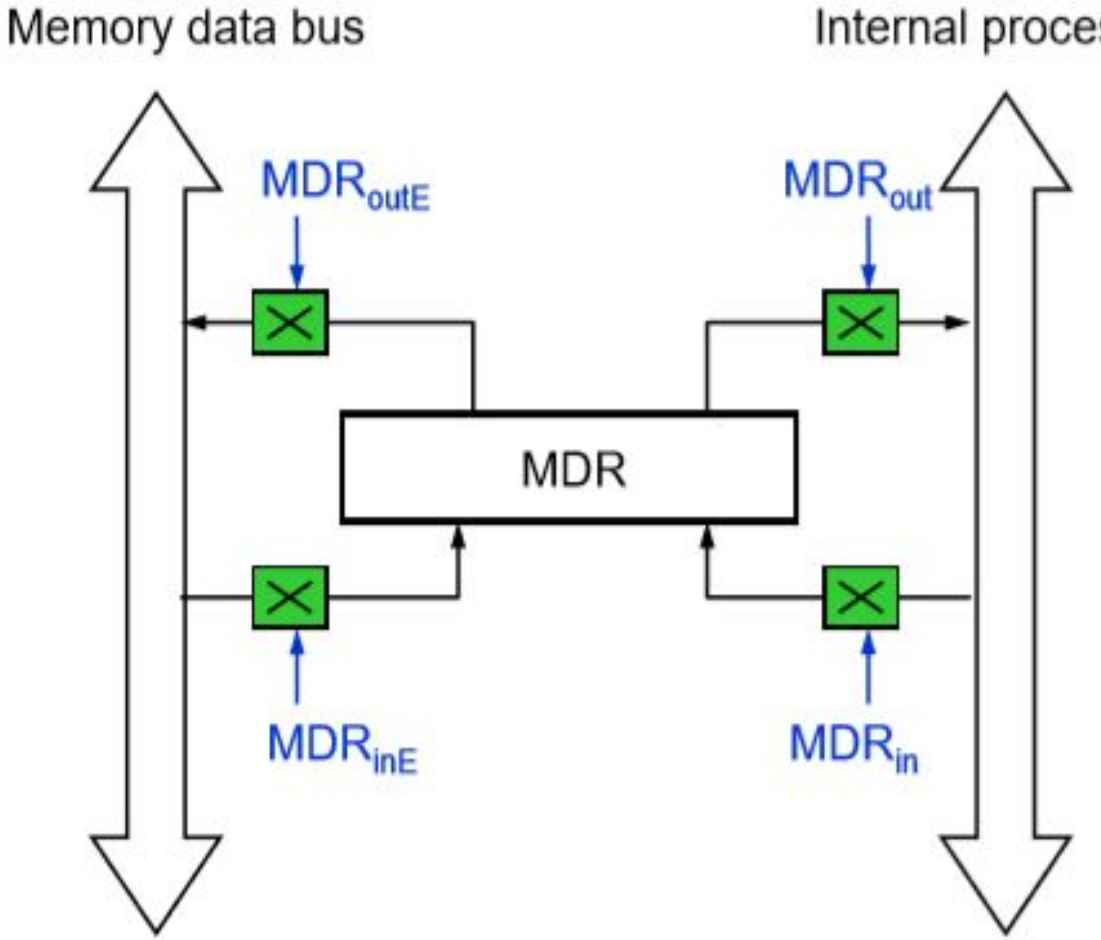


Arithmetic Operation

Add R3, R1, R2



Fetching a Word from Memory



Move (R1), R2

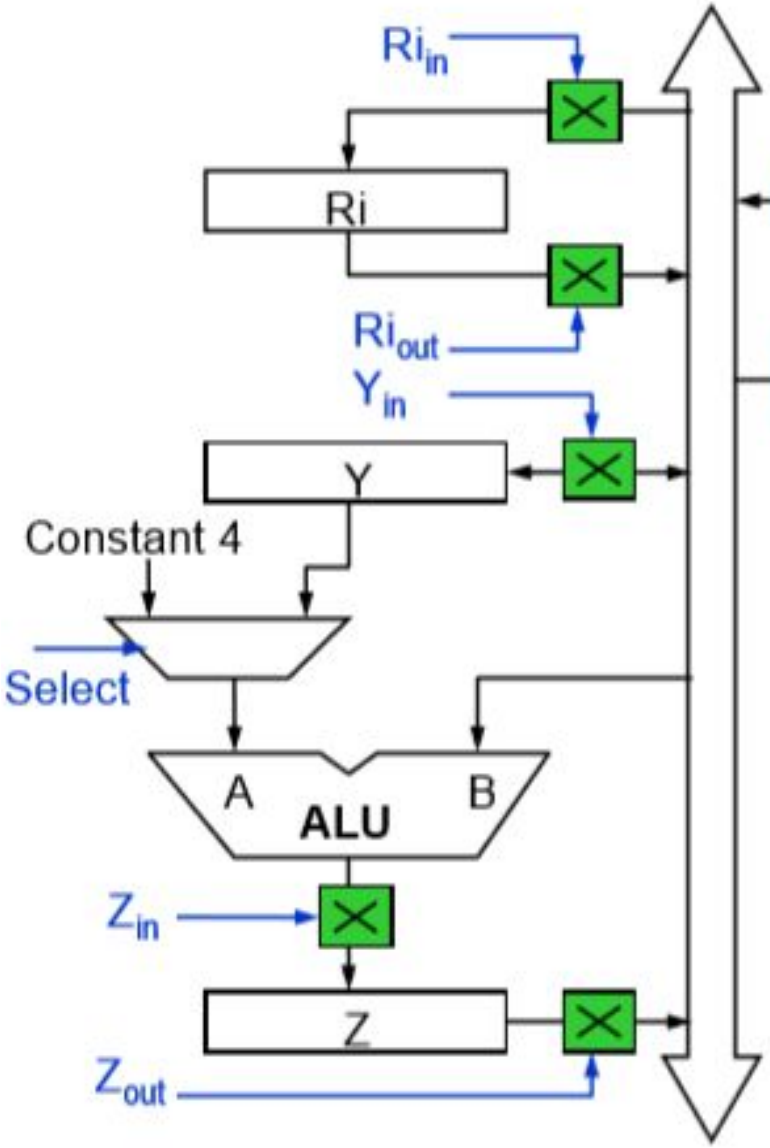
1. R1$_{out}$, MARin, Read
2. MDRinE, WMFC
3. MDRout, R2in

WMFC: is the control signal that causes the processor's control circuitry to wait for the arrival of the MFC signal.

Execution of a Complete Instruction

- Ass (R3), R1: adds the contents of a memory location pointed to by R3 to register R1. Executing this instruction requires the following actions:
  - Fetch the instruction
  - Fetch the first operand (the contents of the memory location pointed to by R3)
  - Perform the addition
  - Load the result into R1

Control Sequence



Instruction fetch

Step	Action
1	PC_{out}, MAR_{in}, Read, Select4 Add, Z_{in}
2	Z_{out}, PC_{in}, Y_{in}, WMFC
3	MDR_{out}, IR_{in}
4	$R3_{out}$, MAR_{in}, Read
5	$R1_{out}$, Y_{in}, WMFC
6	MDR_{out}, SelectY, Add, Z_{in}
7	Z_{out}, $R1_{in}$, End

- JZ 2000H

Control

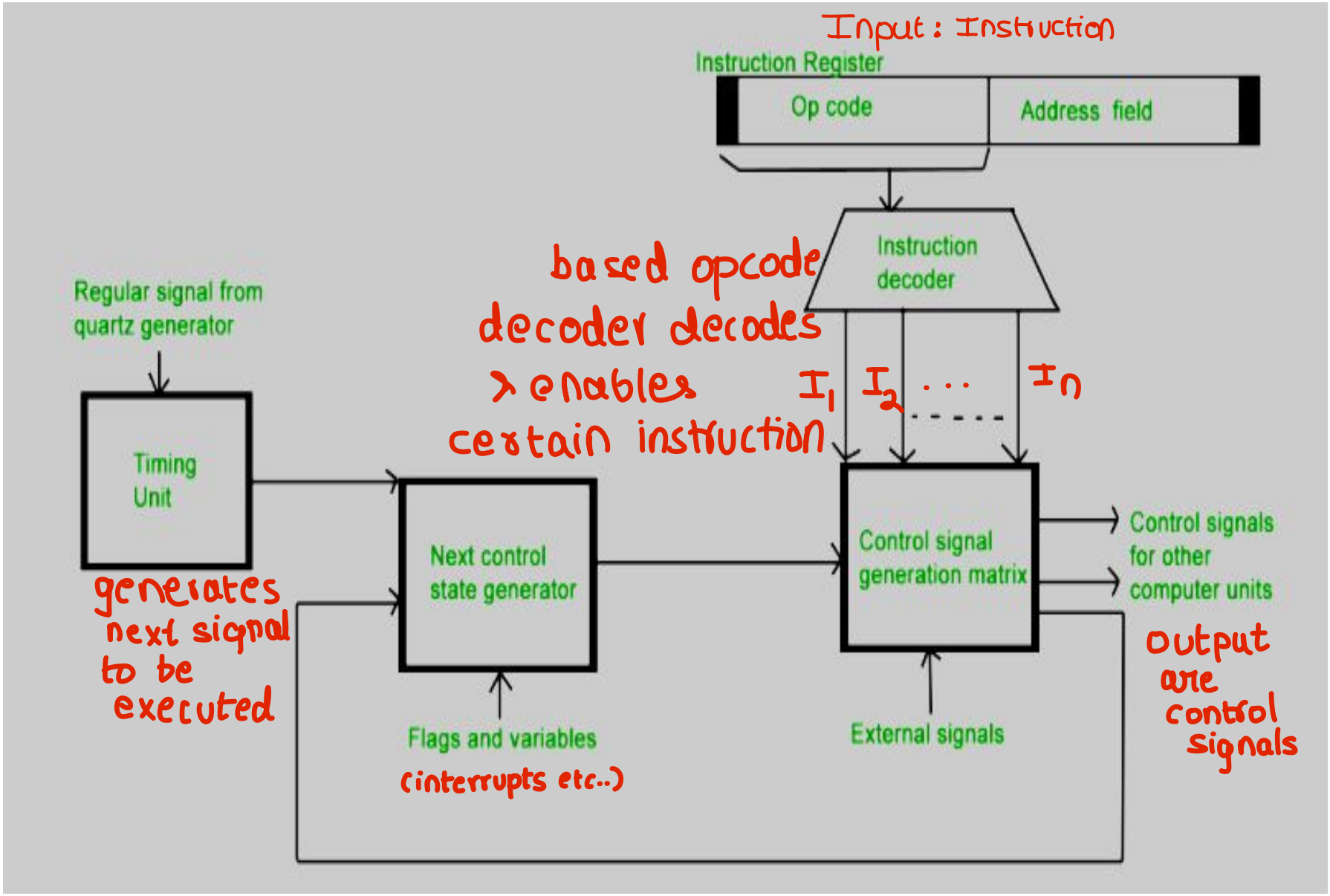
- To execute instructions, the processor must have some means of generating the control signals needed in the proper sequence. Computer designers use a wide variety of techniques to solve this problem.
- The approaches used fall into one of two categories: hardwired control and microprogrammed control

Hardwired Control Unit –

The control hardware can be viewed as a state machine that changes from one state to another in every clock cycle, depending on the contents of the instruction register, the condition codes and the external inputs. The outputs of the state machine are the control signals. The sequence of the operation carried out by this machine is determined by the wiring of the logic elements and hence named as "hardwired".

- Fixed logic circuits that correspond directly to the Boolean expressions are used to generate the control signals.
- Hardwired control is faster than micro-programmed control.
- A controller that uses this approach can operate at high speed.
- RISC architecture is based on hardwired control unit

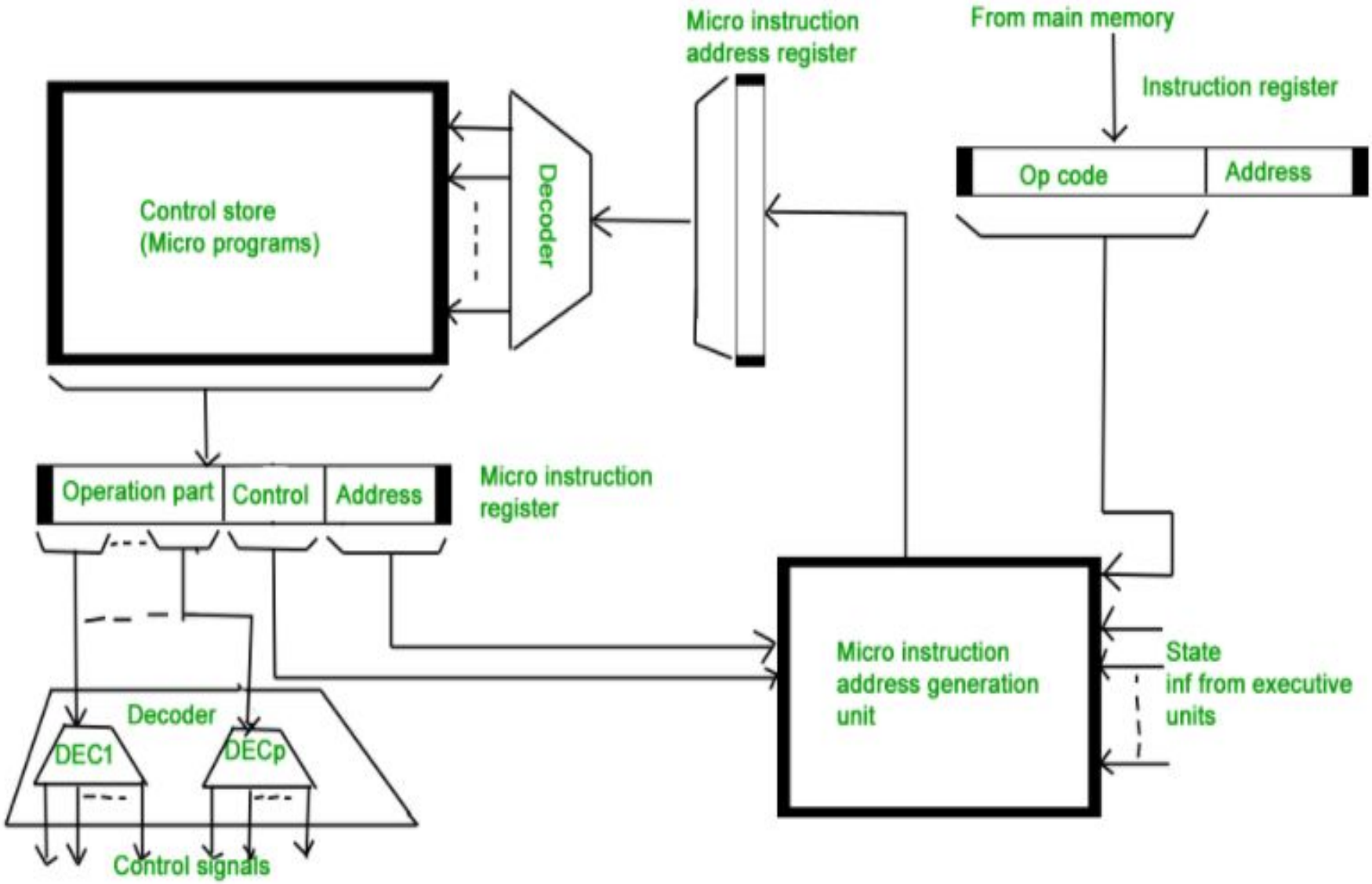
* fast
* Used in RISC


Input : Instruction
Instruction Register
Op code
Address field
Instruction decoder
based opcode decoder decodes > enables certain instruction
I_1 I_2 ... I_n
Regular signal from quartz generator
Timing Unit
generates next signal to be executed
Next control state generator
Flags and variables (interrupts etc..)
Control signal generation matrix
External signals
Control signals for other computer units
Output are control signals

Micro-programmed Control Unit –

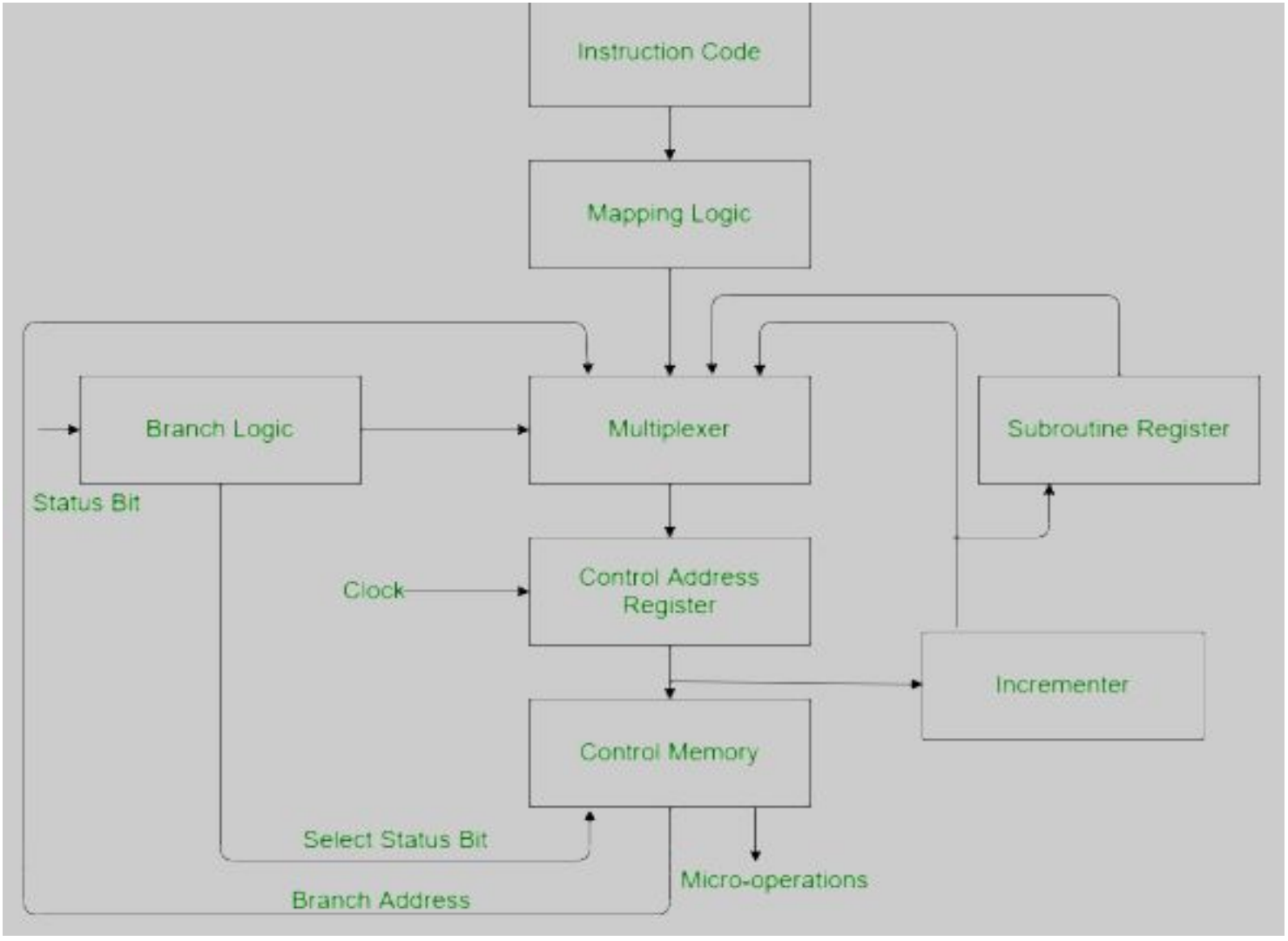
- The control signals associated with operations are stored in special memory units inaccessible by the programmer as Control Words.
- Control signals are generated by a program are similar to machine language programs.
- Micro-programmed control unit is slower in speed because of the time it takes to fetch microinstructions from the control memory.

1. **Control Word :** A control word is a word whose individual bits represent various control signals.
2. **Micro-routine :** A sequence of control words corresponding to the control sequence of a machine instruction constitutes the micro-routine for that instruction.
3. **Micro-instruction :** Individual control words in this micro-routine are referred to as microinstructions.
4. **Micro-program :** A sequence of micro-instructions is called a micro-program, which is stored in a ROM or RAM called a Control Memory (CM).
5. **Control Store :** the micro-routines for all instructions in the instruction set of a computer are stored in a special memory called the Control Store.


Micro instruction address register
From main memory
Instruction register
Op code
Address
Control store (Micro programs)
Decoder
Operation part
Control
Address
Micro instruction register
Micro instruction address generation unit
State inf from executive units
Decoder
DEC1
DECp
Control signals

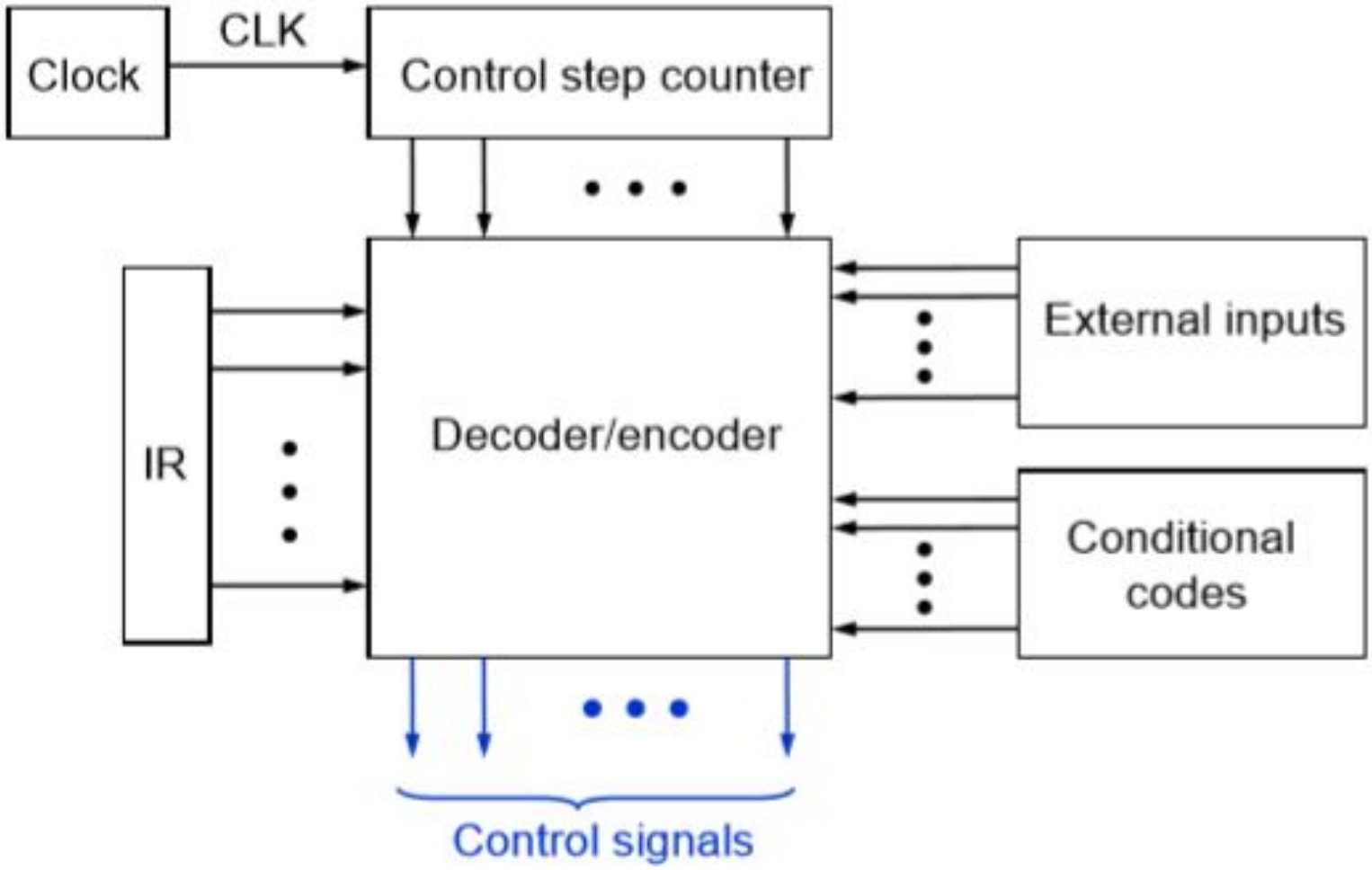
Hardwired Control Unit	Microprogrammed Control Unit
Hardwired control unit generates the control signals needed for the processor using logic circuits	Micrprogrammed control unit generates the control signals with the help of micro instructions stored in control memory
Hardwired control unit is faster when compared to microprogrammed control unit as the required control signals are generated with the help of hardwares	This is slower than the other as micro instructions are used for generating signals here
Difficult to modify as the control signals that need to be generated are hard wired	Easy to modify as the modification need to be done only at the instruction level
More costlier as everything has to be realized in terms of logic gates	Less costlier than hardwired control as only micro instructions are used for generating control signals
It cannot handle complex instructions as the circuit design for it becomes complex	It can handle complex instructions
Only limited number of instructions are used due to the hardware implementation	Control signals for many instructions can be generated
Used in computer that makes use of Reduced Instruction Set Computers(RISC)	Used in computer that makes use of Complex Instruction Set Computers(CISC)

Micro Instructions Sequencer is a combination of all hardware for selecting the next micro-instruction address. The micro-instruction in control memory contains a set of bits to initiate micro operations in computer registers and other bits to specify the method by which the address is obtained.



- **Control Address Register(CAR) :**
  Control address register receives the address from four different paths. For receiving the addresses from four different paths, Multiplexer is used.
- **Multiplexer :**
  Multiplexer is a combinational circuit which contains many data inputs and single data output depending on control or select inputs.
- **Branching :**
  Branching is achieved by specifying the branch address in one of the fields of the micro instruction. Conditional branching is obtained by using part of the micro-instruction to select a specific status bit in order to determine its condition.
- **Mapping Logic :**
  An external address is transferred into control memory via a mapping logic circuit.
- **Incrementer :**
  Incrementer increments the content of the control address register by one, to select the next micro-instruction in sequence.
- **Subroutine Register (SBR) :**
  The return address for a subroutine is stored in a special register called Subroutine Register whose value is then used when the micro-program wishes to return from the subroutine.
- **Control Memory :**
  Control memory is a type of memory which contains addressable storage registers. Data is temporarily stored in control memory. Control memory can be accessed quicker than main memory.

Control Unit Organization



Detailed Block Description

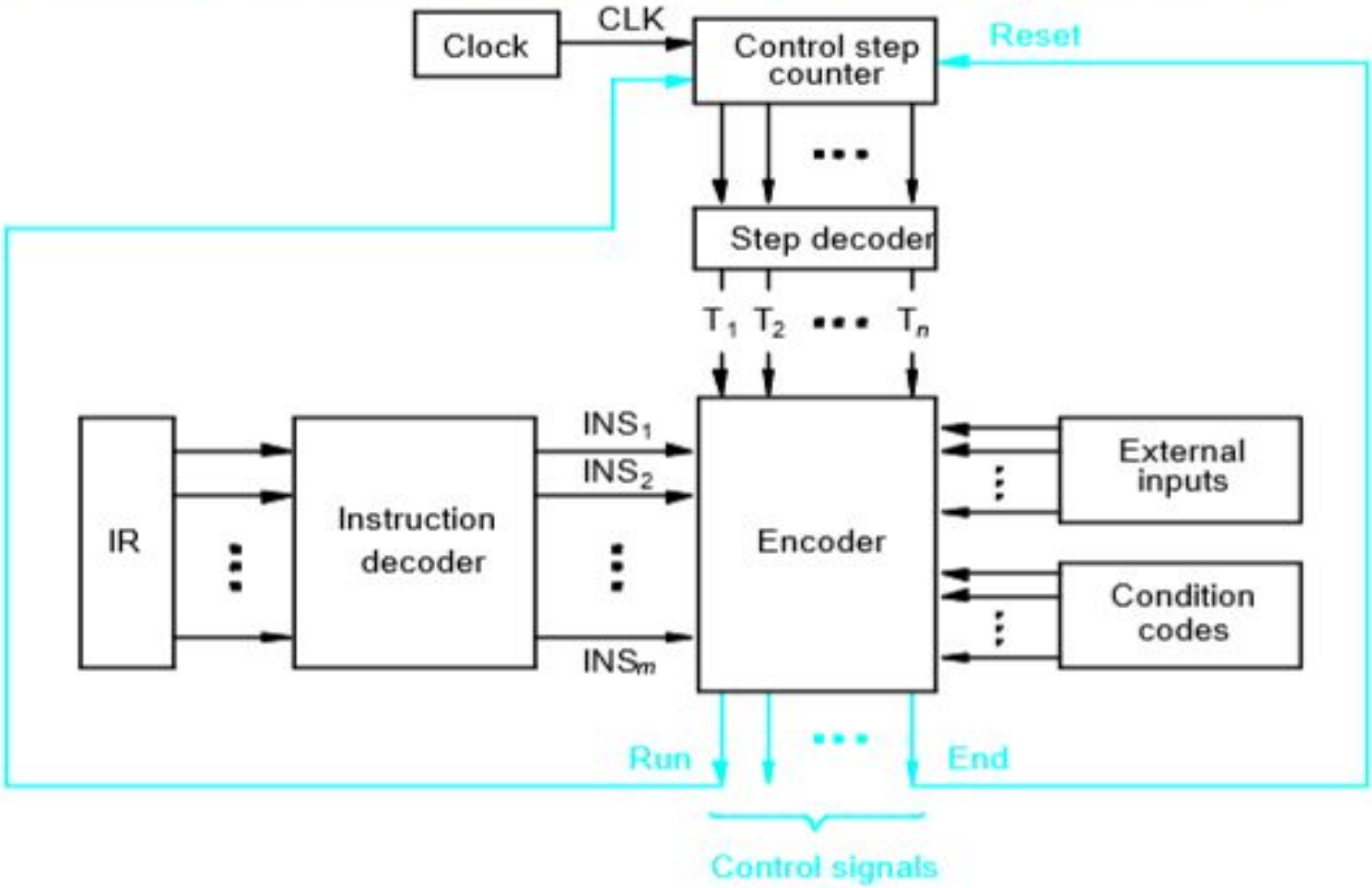
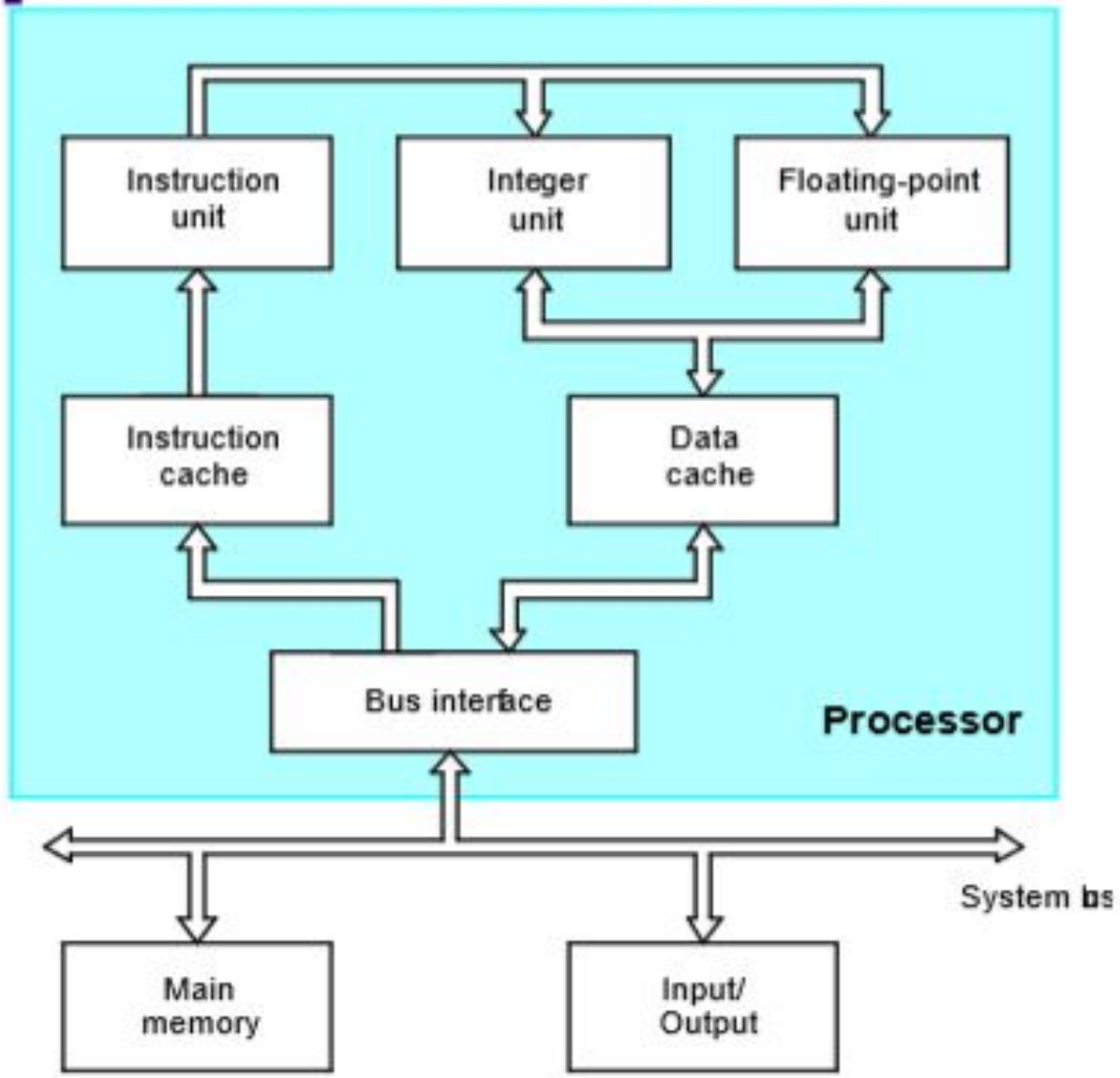


Figure 7.11. Separation of the decoding and encoding functio

A Complete Processor



Microinstructions

- A straightforward way to structure microinstructions is to assign one bit position to each control signal.
- However, this is very inefficient.
- The length can be reduced: most signals are not needed simultaneously, and many signals are mutually exclusive.
- All mutually exclusive signals are placed in the same group in binary coding.

- Control signals are generated by a program similar to machine language programs.
- Control Word (CW); microroutine; microinstruction

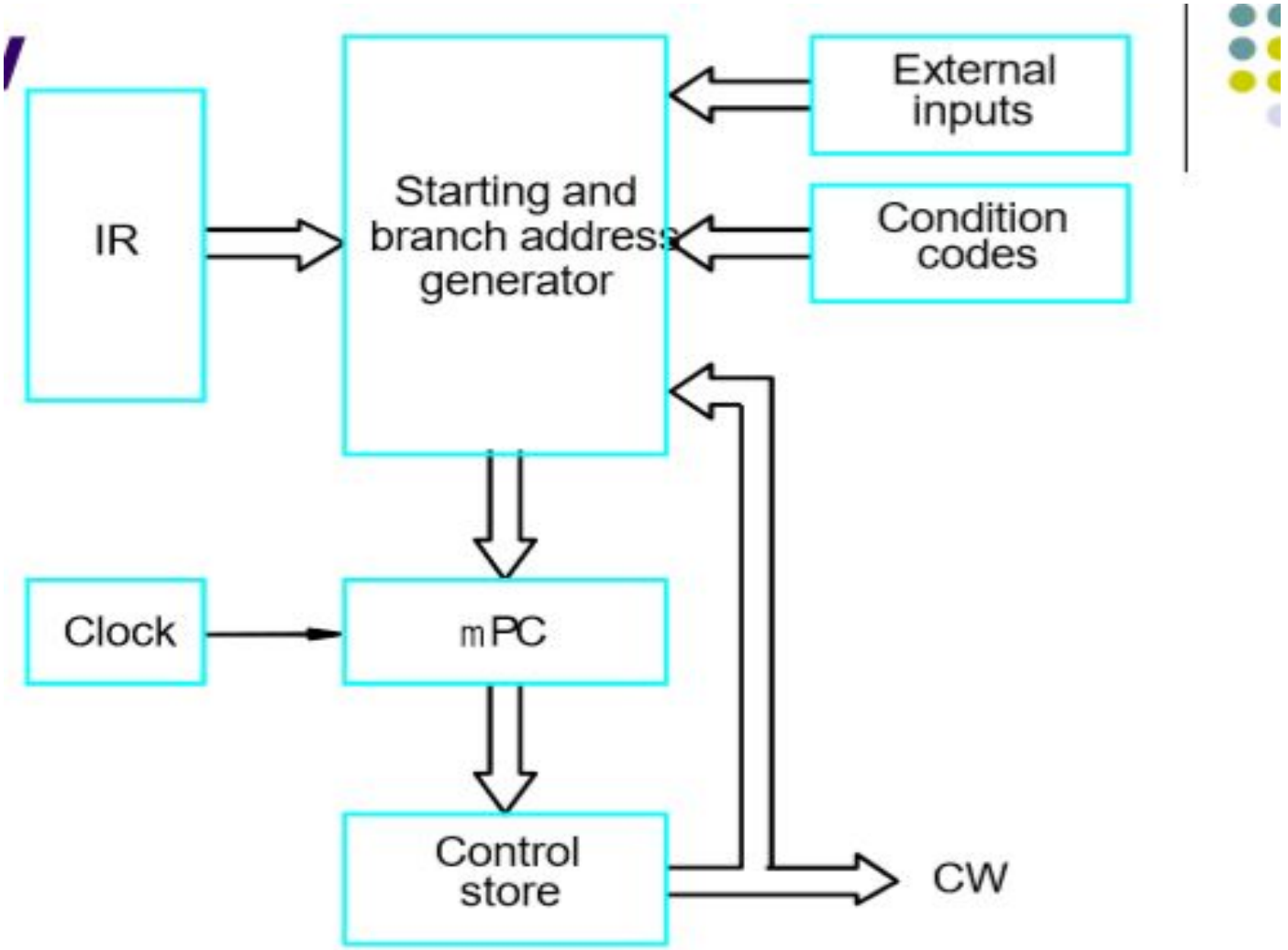
Micro - instruction	:	PC_{in}	PC_{out}	MAR_{in}	Read	MDR_{out}	IR_{in}	Y_{in}	Select	Add	Z_{in}	Z_{out}	$R1_{out}$	$R1_{in}$	$R3_{out}$	WMFC	End	;
1		0	1	1	1	0	0	0	1	1	1	0	0	0	0	0	0	
2		1	0	0	0	0	0	1	0	0	0	1	0	0	0	1	0	
3		0	0	0	0	1	1	0	0	0	0	0	0	0	0	0	0	
4		0	0	1	1	0	0	0	0	0	0	0	0	0	1	0	0	
5		0	0	0	0	0	0	1	0	0	0	0	1	0	0	1	0	
6		0	0	0	0	1	0	0	0	1	1	0	0	0	0	0	0	
7		0	0	0	0	0	0	0	0	0	0	1	0	1	0	0	1	

Micro-programmed control unit can be classified into two types based on the type of Control Word stored in the Control Memory, viz., Horizontal micro-programmed control unit and Vertical micro-programmed control unit.

- In *Horizontal micro-programmed* control unit, the control signals are represented in the decoded binary format, i.e., 1 bit/CS. Here 'n' control signals require n bit encoding. On the other hand.
- In *Vertical micro-programmed* control unit, the control signals are represented in the encoded binary format. Here 'n' control signals require $\log_2 n$ bit encoding.

Horizontal μ-programmed CU	Vertical μ-programmed CU
It supports longer control word.	It supports shorter control word.
It allows higher degree of parallelism. If degree is n, then n Control Signals are enabled at a time.	It allows low degree of parallelism i.e., degree of parallelism is either 0 or 1.
No additional hardware is required.	Additional hardware in the form of decoders are required to generate control signals.

It is faster than Vertical micro-programmed control unit.	it is slower than Horizontal micro-programmed control unit.
It is less flexible than Vertical micro-programmed control unit.	It is more flexible than Horizontal micro-programmed control unit.
Horizontal micro-programmed control unit uses horizontal microinstruction, where every bit in the control field attaches to a control line.	Vertical micro-programmed control unit uses vertical microinstruction, where a code is used for each action to be performedand thedecoder translates this code into individual control signals.
Horizontal micro-programmed control unit makes less use of ROM encoding than vertical micro-programmed control unit.	Vertical micro-programmed control unit makes more use of ROM encoding to reduce the length of the control word.



Organization of the control unit to allow conditional branching in the microprogram.

Microinstructions with the next-address field.

- Several branch microinstructions are required to enable sharing of common code.
- The branch microinstructions do not perform any useful operation related to data.
- They are required to determine the address of the next microinstruction.
- They slow down the execution of the instruction.

- Ideally we need to assign consecutive addresses to all microinstructions that are generally executed one after the other.
- Recall that the next microinstruction is determined by incrementing the microprogram counter.
- But due to the goal of sharing as much common code as possible, this is not always possible.
- This leads to a significant increase in the branch instructions.

Microinstructions with the next-address field.

- Powerful alternative is to include an address field as a part of every microinstruction.
- The address field indicates the location of the next microinstruction to be fetched.
- In effect, every microinstruction becomes a branch microinstruction in addition to its other function.

Disadvantages:

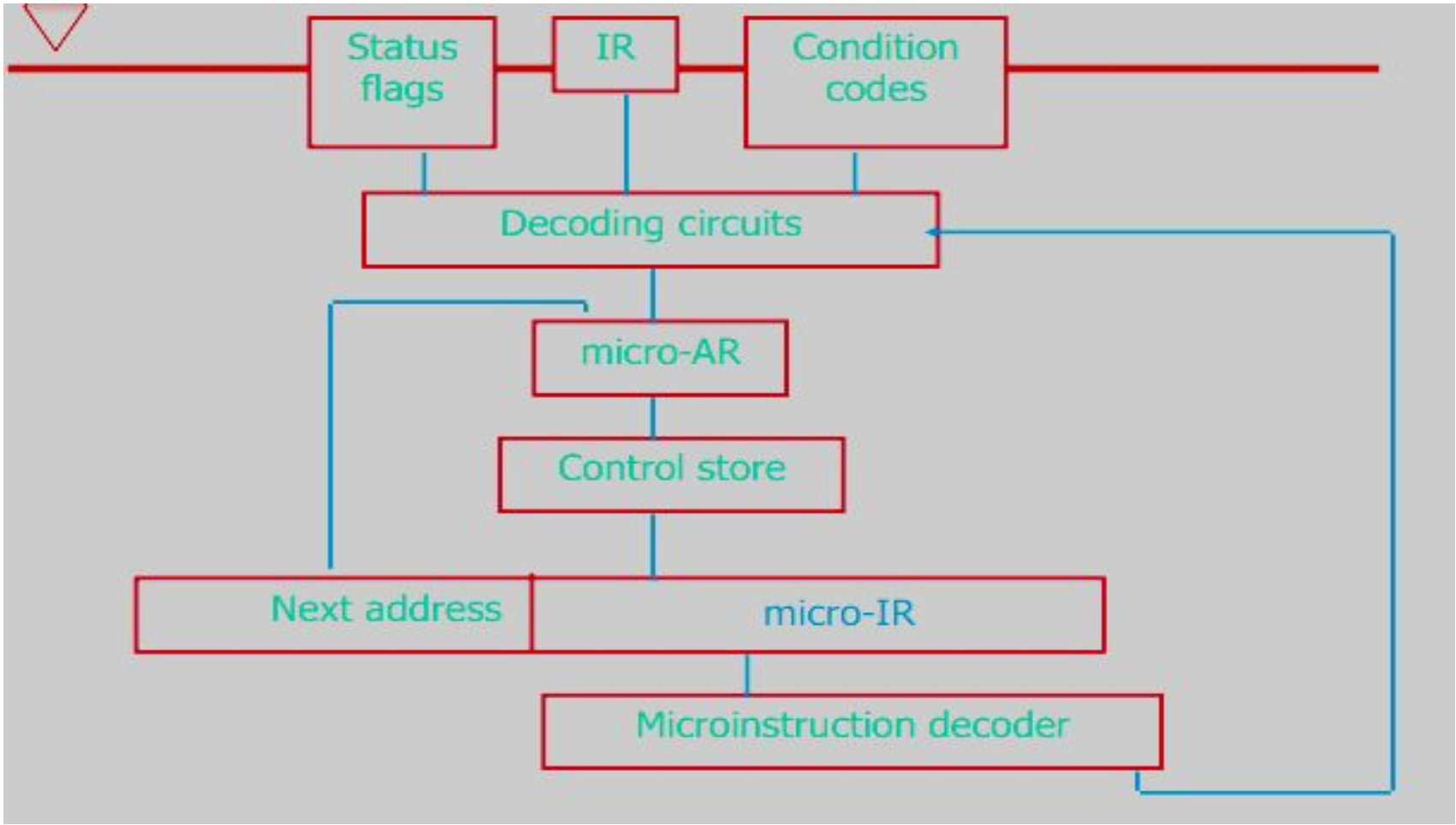
- Additional bits are required to specify the address field in every instruction.
- Approximately one-sixth of the control store is devoted to specifying the address.

Advantages:

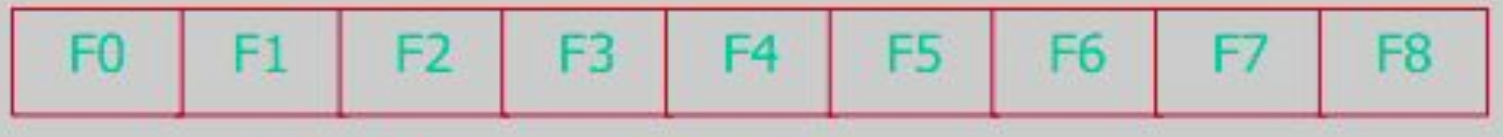
- Separate branch instructions are virtually eliminated.
- Flexible scheme, very few restrictions in assigning addresses to microinstructions.

Next Address Field

- Micro-instruction contains address next micro-instruction
- Larger store needed
- Branch micro-instructions no longer needed



- The microprogram discussed earlier requires several branch microinstructions, which perform no useful operation in the datapath.
- A powerful alternative approach is to include an address field as a part of every microinstruction to indicate the location of the next microinstruction to be fetched.
- Pros: separate branch microinstructions are virtually eliminated; few limitations in assigning addresses to microinstructions.
- Cons: additional bits for the address field (around 1/6)



Field 0(8 bits): Next address
Field 1(4 bits): Register address_in
Field 2(4 bits): Register address_out
Field 3(4 bits): Other registers_in
Field 4(4 bits): Function ALU
Field 5(2 bit) : Read/Write/Nop
Field 6(1 bit) : Carry-in ALU
Field 7(1 bit) : WMFC
Field 8(1 bit) : End
..... PLA/ORing etc

Microinstruction

F1 (4 bits)	F2 (3 bits)	F3 (3 bits)	F4 (4 bits)	F5 (2 bits)
0000: No transfer	000: No transfer	000: No transfer	0000: Add	00: No action
0001: PC_{out}	001: PC_{in}	001: MAR_{in}	0001: Sub	01: Read
0010: MDR_{out}	010: IR_{in}	010: MDR_{in}	⋮	10: Write
0011: Z_{out}	011: Z_{in}	011: $TEMP_{in}$	1111: XOR	
0100: $R0_{out}$	100: $R0_{in}$	100: Y_{in}	16 ALU functions	
0101: $R1_{out}$	101: $R1_{in}$			
0110: $R2_{out}$	110: $R2_{in}$			
0111: $R3_{out}$	111: $R3_{in}$			
1010: $TEMP_{out}$				
1011: $Offset_{out}$				

F6 (1 bit)	F7 (1 bit)	F8 (1 bit)
0: SelectY	0: No action	0: Continue
1: Select4	1: WMFC	1: End

What is the price paid for this scheme?

- *Each group occupies a large enough field to represent all the signals.*
- *Most fields must include one inactive code, which specifies no action.*
- *All fields do not have to include inactive code.*

Require a little more hardware