SNS COLLEGE OF ENGINEERING

Kurumbapalayam (Po), Coimbatore – 641 107

An Autonomous Institution

Accredited by NBA – AICTE and Accredited by NAAC – UGC with 'A' Grade
Approved by AICTE, New Delhi & Affiliated to Anna University, Chennai

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

COURSE NAME :19IT301 COMPUTER ORGANIZATION AND ARCHITECTURE

II YEAR /III SEMESTER

Unit 3- **PROCESSOR AND PIPELINING**

Topics 1 to 5 : Fundamental concepts – Execution of a complete instruction – Multiple bus organization – Hardwired control – Micro programmed control

UNIT 3 PROCESSOR AND PIPELINING

Fundamental concepts – Execution of a complete instruction – Multiple bus organization – Hardwired control – Micro programmed control – Pipelining: Basic concepts – Data hazards – Instruction hazards – Influence on Instruction sets – Data path and control consideration.

FUNDAMENTAL CONCEPTS

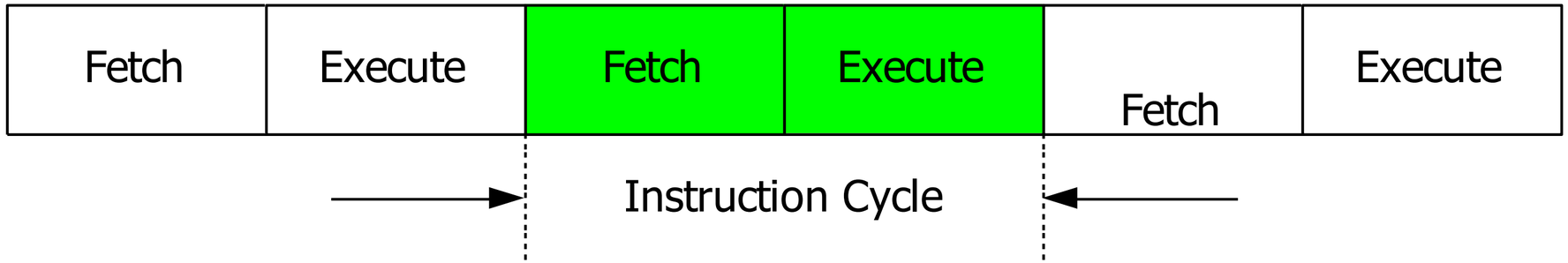
Introduction to CPU

The operation or task that must perform by CPU are:

- ✓**Fetch Instruction:** The CPU reads an instruction from memory.
- ✓**Interpret Instruction:** The instruction is decoded to determine what action is required.
- ✓**Fetch Data:** The execution of an instruction may require reading data from memory or I/O module.
- ✓**Process data:** The execution of an instruction may require performing some arithmetic or logical operation on data.
- ✓**Write data:** The result of an execution may require writing data to memory or an I/O module.

CPU : Basic operations

Fetch : Read the instructions and data from memory
Execute : perform the desired operation and write the result into the memory or registers



Concept of Program Execution

Execution of one instruction requires the following three steps to be performed by the CPU:

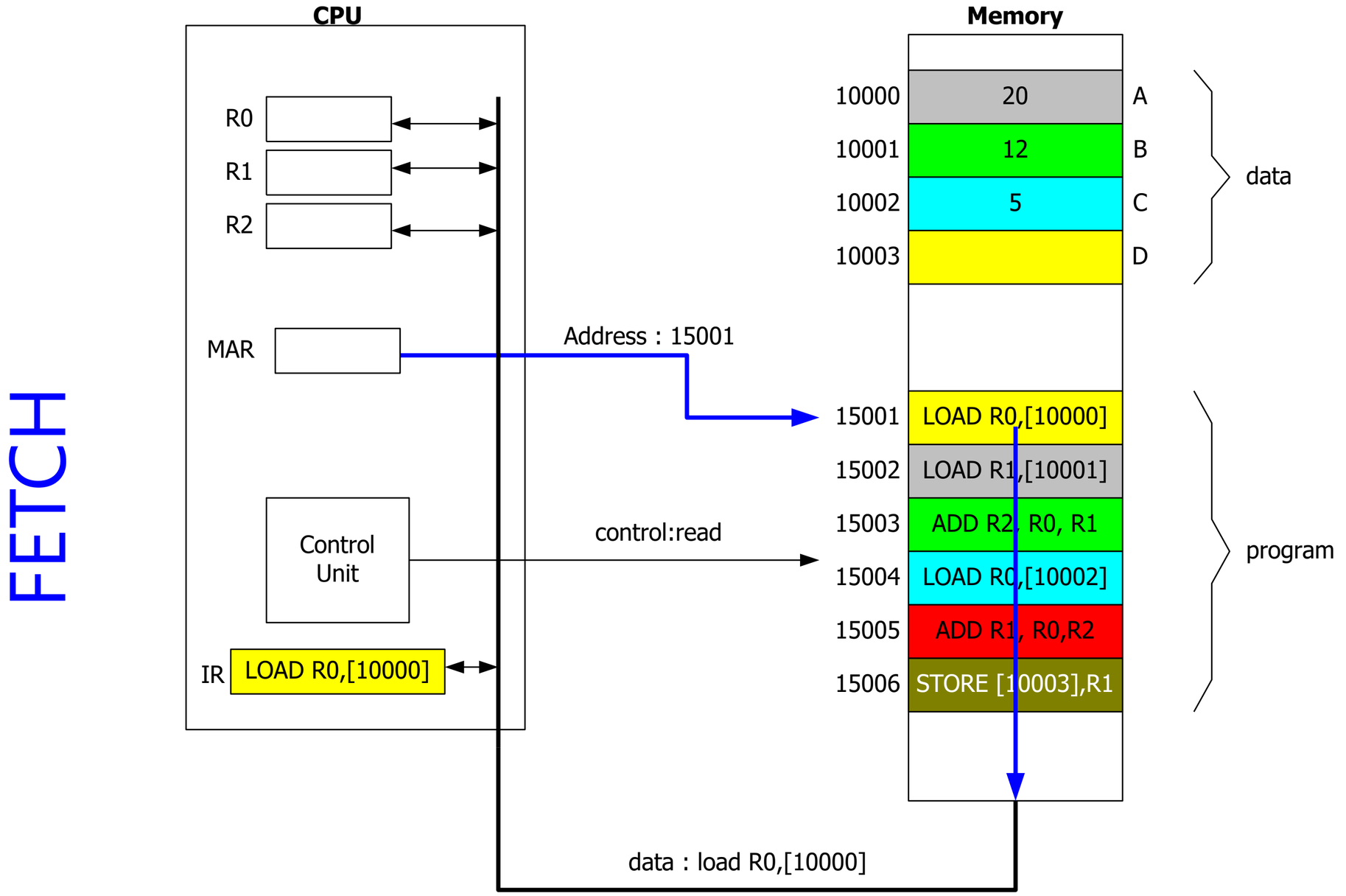
1.Fetch the contents of the memory location pointed at by the PC. The contents of this location are interpreted as an instruction to be executed. Hence, they are stored in the instruction register (IR). Symbolically this can be written as:

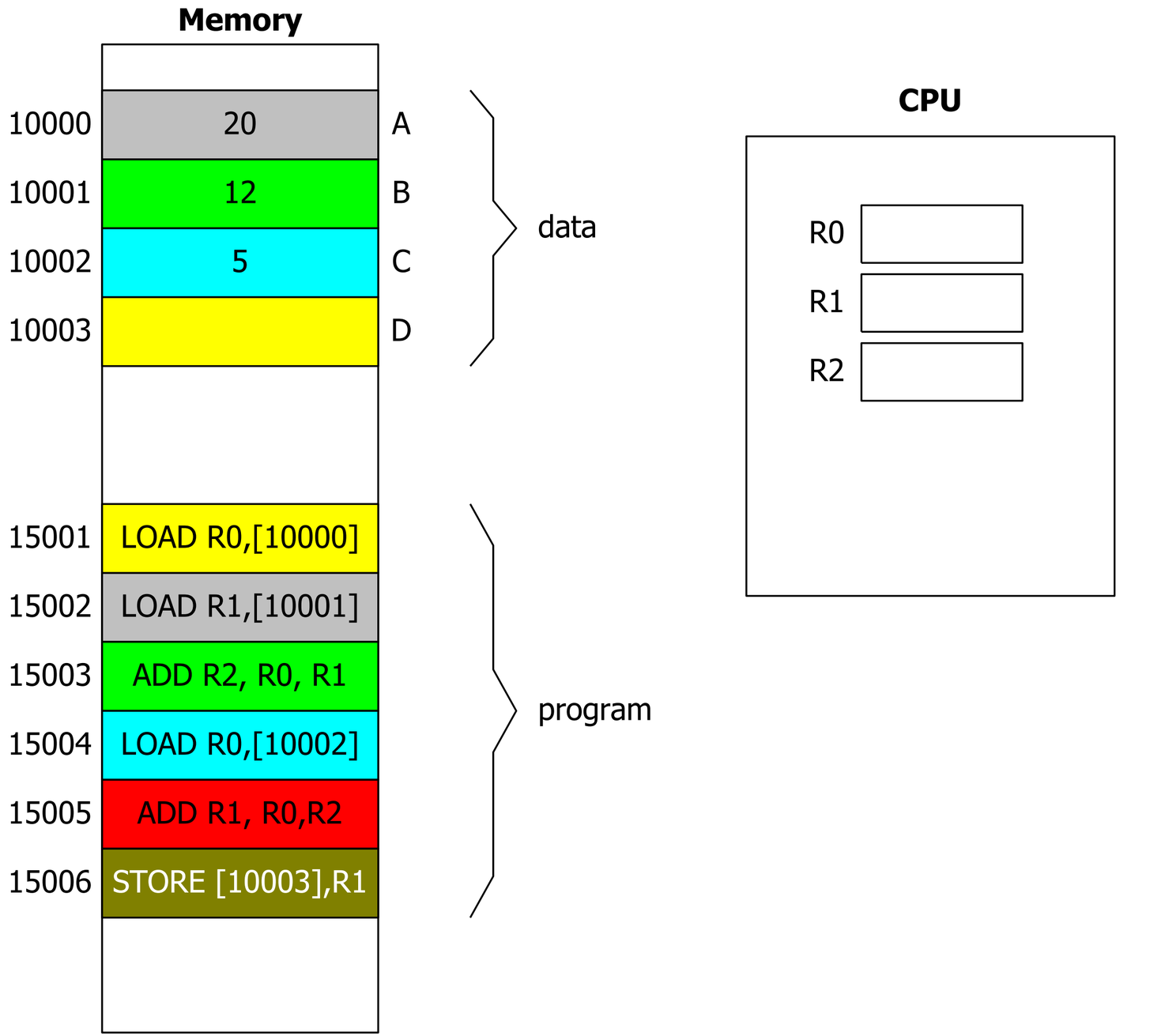
IR = [[PC]]

2.Increment the contents of the PC by 4.

PC = [PC] + 4

3.Carry out the actions specified by the instruction stored


FETCH
CPU
R0
R1
R2
MAR
Address : 15001
Control Unit
control:read
IR
LOAD R0,[10000]
Memory
10000
20
A
10001
12
B
10002
5
C
10003
D
data
15001
LOAD R0,[10000]
15002
LOAD R1,[10001]
15003
ADD R2, R0, R1
15004
LOAD R0,[10002]
15005
ADD R1, R0,R2
15006
STORE [10003],R1
program
data : load R0,[10000]


Memory
10000
20
A
10001
12
B
10002
5
C
10003
D
data
15001
LOAD R0,[10000]
15002
LOAD R1,[10001]
15003
ADD R2, R0, R1
15004
LOAD R0,[10002]
15005
ADD R1, R0,R2
15006
STORE [10003],R1
program
CPU
R0
R1
R2

$MOVE\ (R2),R1$

As an example, assume that the address of the memory location to be accessed is kept in register R2 and that the memory contents to be loaded into register R1. This is done by the following sequence of operations:

1. MAR [R2]
2. Read
3. Wait for MFC signal
4. R1 [MDR]

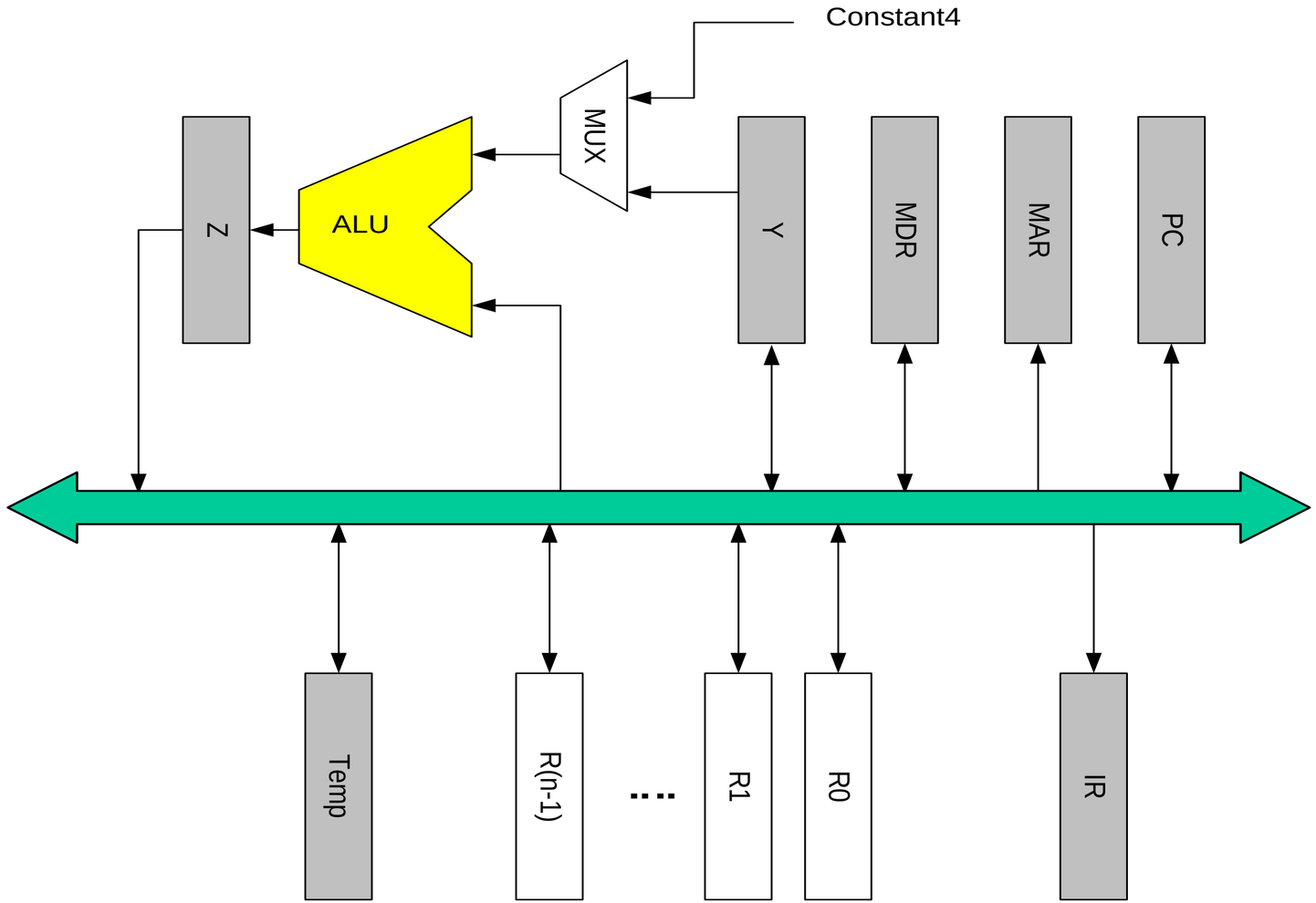
The time required for step 3 depends on the speed of the memory unit. In general, the time required to access a word from the memory is longer than the time required to perform any operation within the CPU.

Instructions of CPU

There are 4 types of instructions

1. Data transfer between memory and CPU registers
2. Arithmetic and Logic Operations on data
3. Program Sequencing and Control
4. I/O transfer

Inside simple CPU with Single-bus Datapath



For the execution of an instruction, we need to perform an instruction cycle. An instruction cycle consists of two phase,

1. Fetch cycle and
2. Execution cycle.

Most of the operation of a CPU can be carried out by performing one or more of the following functions in some specified sequence:

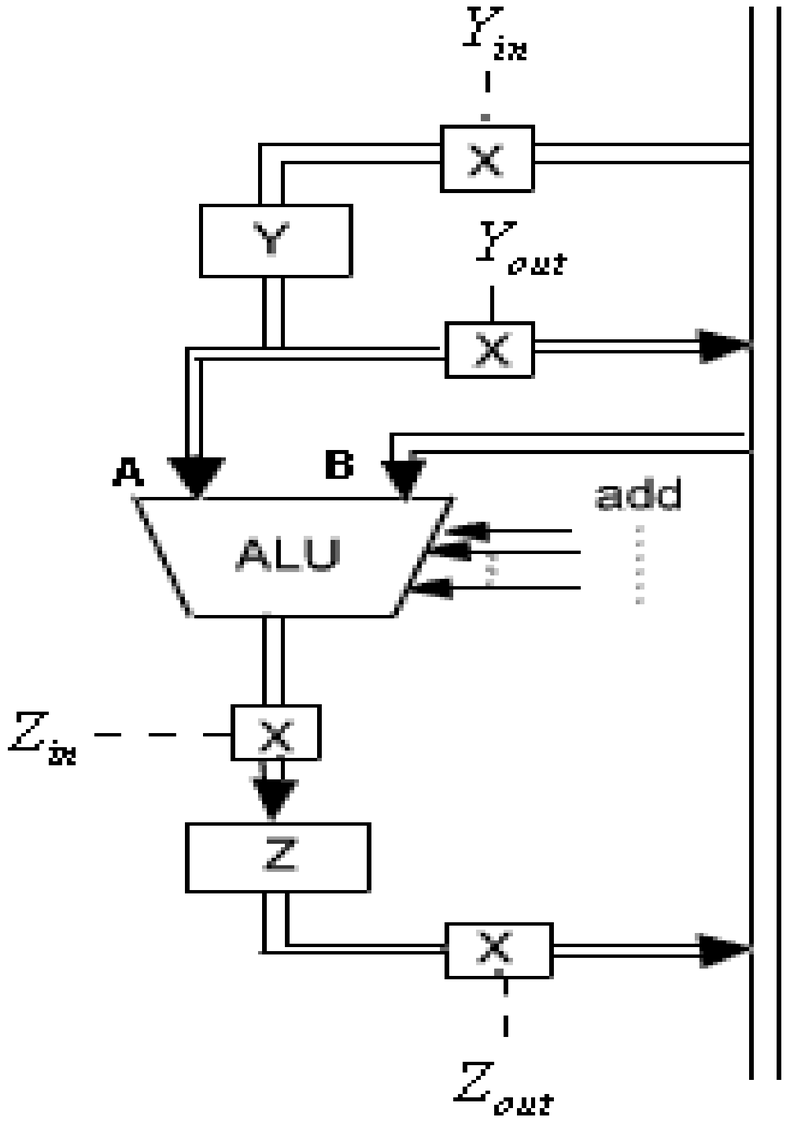
✓Fetch the contents of a given memory location and load them into a CPU register.
✓Store a word of data from a CPU register into a given memory location.
✓Transfer a word of data from one CPU register to another or to the ALU.
✓Perform an arithmetic or logic operation, and store the result in a CPU register.

Register Transfer

- ✓ Example, to transfer the contents of register R1 to register R2, the following actions are needed:

- ✓ Enable the output gate of register R1 by setting R1out to 1.
  -- This places the contents of R1 on the CPU bus.
- ✓ Enable the input gate of register R2 by setting R2 in to 1.
  -- This loads data from the CPU bus into the register R2.

- ✓ **Performing the arithmetic or logic operation:**
- ✓ To perform any arithmetic or logic operation (say binary operation) both the input should be made available at the two inputs of the ALU simultaneously.
- ✓ Once both the inputs are available then appropriate signal is generated to perform the required operation. Use temporary storage (register) to carry out the operation in ALU .
- ✓ The sequence of operations that have to carried out to perform one ALU operation depends on the organization of the CPU. Consider an organization in which one of the operand of ALU is stored in some temporary register Y and other operand is directly taken from CPU internal bus.
- ✓ The result of the ALU operation is stored in another temporary register Z. This organization is shown in the Figure


Y_{in}
X
Y
Y_{out}
X
A
B
add
ALU
Z_{in}
X
Z
X
Z_{out}

$ADD\ R1,R2,R3$

Therefore, the sequence of operations to add the contents of register R1 to register R2 and store the result in register R3 should be as follows:

R1out, Yin
R2out, Add, Zin
Zout, R3in

Storing a word into memory

The procedure of writing a word into memory location is similar to that for reading one from memory.

The only difference is that the data word to be written is first loaded into the MDR, the write command is issued.

Example MOVE R1,(R2) The memory write operation requires the following sequence:

MAR [R2]
MDR [R1]
Write
Wait for MFC

1. R1out,MARin
2. R2out,MDRin,write
3. MDRout,WMFC

FETCHING A WORD FROM MEMORY

MOVE (R1),R2

ACTIONS:

1.MAR [R1]

2.READ ←

3.WMFC

4.LOAD MDR FROM THE MEMORY BUS

5.R2 [MDR] ←

CONTROL SEQUENCE(ACTIVE SIGNALS)

R1out,MARin,read

MDRin,WMFC

MDRout,R2in

Execution of a Complete Instructions

We have discussed about four different types of basic operations:

- Fetch information from memory to CPU
- Store information from CPU register to memory
- Transfer of data between CPU registers.
- Perform arithmetic or logic operation and store the result in CPU registers.

ADD (R3),R1

Execution of this instruction requires the following action :

- Fetch instruction
- Fetch first operand (Contents of memory location pointed at by R2 of the instruction)
- Perform addition
- Load the result into R1.

ADD (R3),R1 Following sequence of control steps are required to implement the above operation for the single-bus architecture that we have discussed in earlier section.

StepsActions

1.PCout, MARin, Read, select4, Add, Zin
2.Zout, PCin, Yin ,Wait For MFC
3.MDRout, IRin
4. R3out, MARin, Read
5.R1out, Yin, Wait for MFC
6.MDRout, select y,Add, Zin
7.Zout, R1in
8.END

Instructruction execution proceeds as follows:

In Step1:

The instruction fetch operation is initiated by loading the contents of the PC into the MAR and sending a read request to memory.

1.Register Transfers

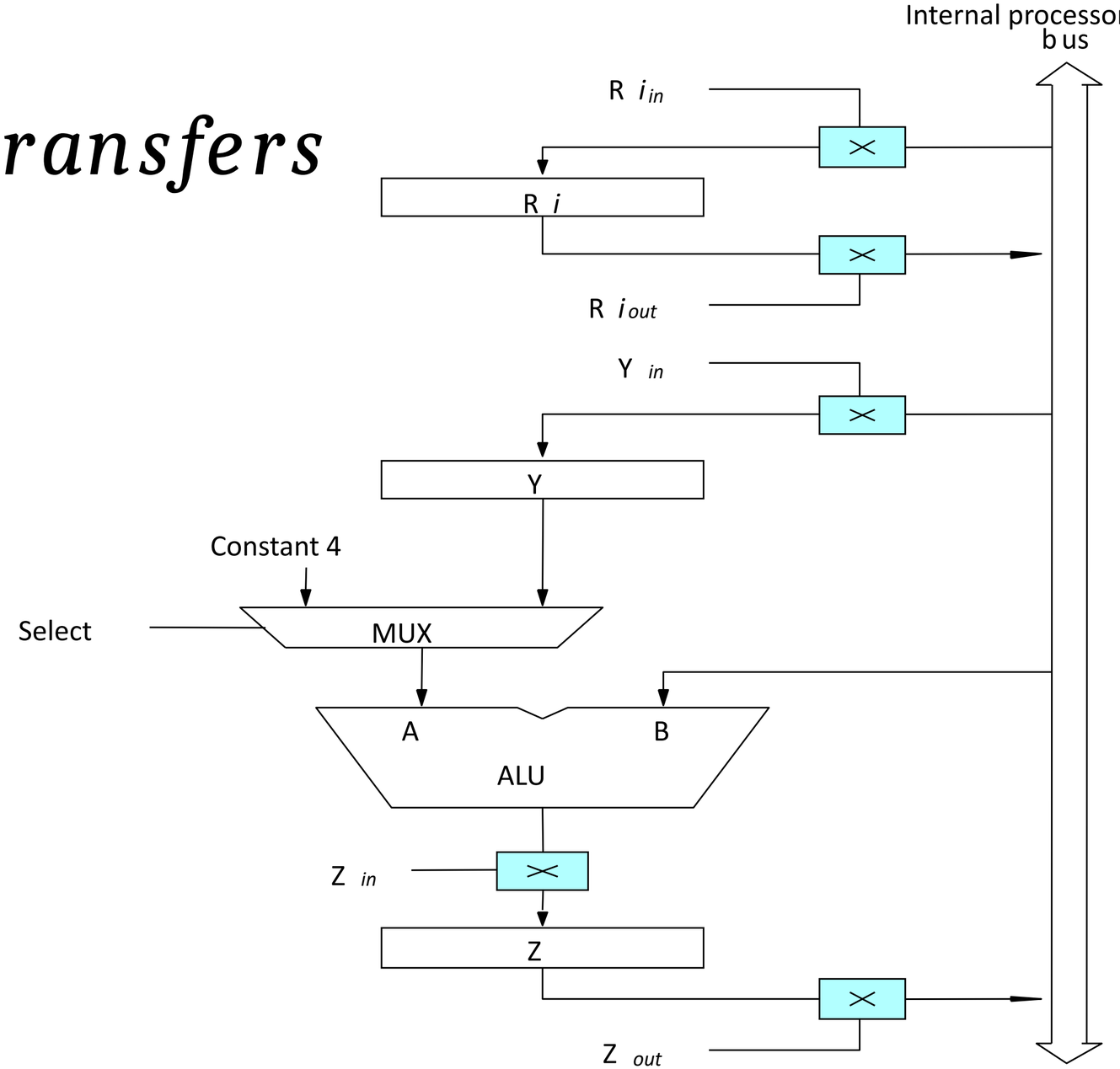


Figure 7.2. Input and output gating for the registers in Figure 7.1.

Execution of a Complete Instruction

Add (R3), R1

To perform this task first of all the contents of PC have to be brought to internal bus and then it is loaded to MAR.To perform this task control circuit has to generate the PCout signal and MARin signal.
After issuing the read signal, CPU has to wait for some time to get the MFC signal. During that time PC is updated by 1 through the use of the ALU. This is accomplished by setting one of the inputs to the ALU (Register Y) to 0 and the other input is available in bus which is current value of PC.
At the same time, the carry-in to the ALU is set to 1 and an add operation is specified.

In Step 2:

The updated value is moved from register Z back into the PC. Step 2 is initiated immediately after issuing the memory Read request without waiting for completion of memory function. This is possible, because step 2 does not use the memory bus and its execution does not depend on the memory read operation.

In Step 3:

Step3 has been delayed until the MFC is received. Once MFC is received, the word fetched from the memory is transfered to IR (Instruction Register), Because it is an instruction. Step 1 through 3 constitute the instruction fetch phase of the control sequence.
The instruction fetch portion is same for all instructions. Next step onwards, instruction execution phase takes place.

As soon as the IR is loaded with instruction, the instruction decoding circuits interprets its contents. This enables the control circuitry to choose the appropriate signals for the remainder of the control sequence, step 4 to 8, which we referred to as the execution phase. To design the control sequence of execution phase, it is needed to have the knowledge of the internal structure and instruction format of the PU. Secondly , the length of instruction phase is different for different instruction.
In this example , we have assumed the following instruction format :
opcode MRi.e., **opcode:** Operation Code

M: Memory address for source

R: Register address for source/destination

In Step 5 :
The destination field of IR, which contains the address of the register R1, is used to transfer the contents of register R1 to register Y and wait for Memory function Complete. When the read operation is completed, the memory operand is available in MDR.

In Step 6 :
The result of addition operation is performed in this step.

In Step 7:
The result of addition operation is transfered from temporary register **Z** to the destination register **R1** in this step.

In step 8 :
It indicates the end of the execution of the instruction by generating End signal. This indicates completion of execution of the current instruction and causes a new fetch cycle to be started by going back to step 1.

Branching

With the help of branching instruction, the control of the execution of the program is transfered from one particular position to some other position, due to which the sequence flow of control is broken. Branching is accomplished by replacing the current contents of the PC by the branch address, that is, the address of the instruction to which branching is required.

Consider a branch instruction in which branch address is obtained by adding an offset **X**, which is given in the address field of the branch instruction, to the current value of PC.

Consider the following unconditional branch instruction

JUMP **X**

i.e., the format is

op- code offset of jump

The control sequence that enables execution of an unconditional branch instruction using the single - bus organization is as follows :

StepsActions

1.PCout, MARin, Read, select4,Add ,Zin
2.Zout, PCin, Wait for MFC
3.MDRout, IRin
4.PCout, Yin
5offset-field-of- IRout,selectY, Add, Zin
6.Zout, PCin
7.End

For unconditional branch
offset-field-of-IRout Add,Zin,If N=0 then End use in step 5

Control sequence for instruction Branch<0

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	Offset-field-of-IR_{out}, ADD, Z_{in}, if N=0 then End
5	Z_{out}, PC_{in}, End

Execution starts as usual with the fetch phase, ending with the instruction being loaded into the IR in step 3. To execute the branch instruction, the execution phase starts in step 4.

In Step 4

The contents of the PC are transferred to register Y.

In Step 5

The offset **X** of the instruction is gated to the bus and the addition operation is performed.

In Step 6

The result of the addition, which represents the branch address is loaded into the PC.

In Step 7

It generates the End signal to indicate the end of execution of the current instruction.

Consider now the conditional branch instruction instead of unconditional branch. In this case, we need to check the status of the condition codes, between step 3 and 4. i.e., before adding the offset value to the PC contents.
For example, if the instruction decoding circuitry interprets the contents of the IR as a branch on Negative(BRN) instruction, the control unit proceeds as follows:First the condition code register is checked. If bit N (negative) is equal to 1 , the control unit proceeds with step 4 trough step 7 of control sequence of unconditional branch instruction.
If , on the other hand , N is equal to 0, and End signal is issued .
This in effect , terminates execution of the branch instruction and causes the instruction immediately following in the branch instruction to be fetched when a new fetch operation is performed.
Therefore , the control sequence for the conditional branch instruction BRN can be obtained from the control sequence of an unconditional branch instruction by replacing the step 4 by

4. If then End
   If N then PCout, yin

Multiple Bus Organization

All register outputs are connected to bus A and B, all register inputs are connected to bus C.

If needed the ALU may simply pass one of its two input operands unmodified to Cbus.ALU control signals R=A,R=B

No need of Y and Z registers

Increment unit is present.

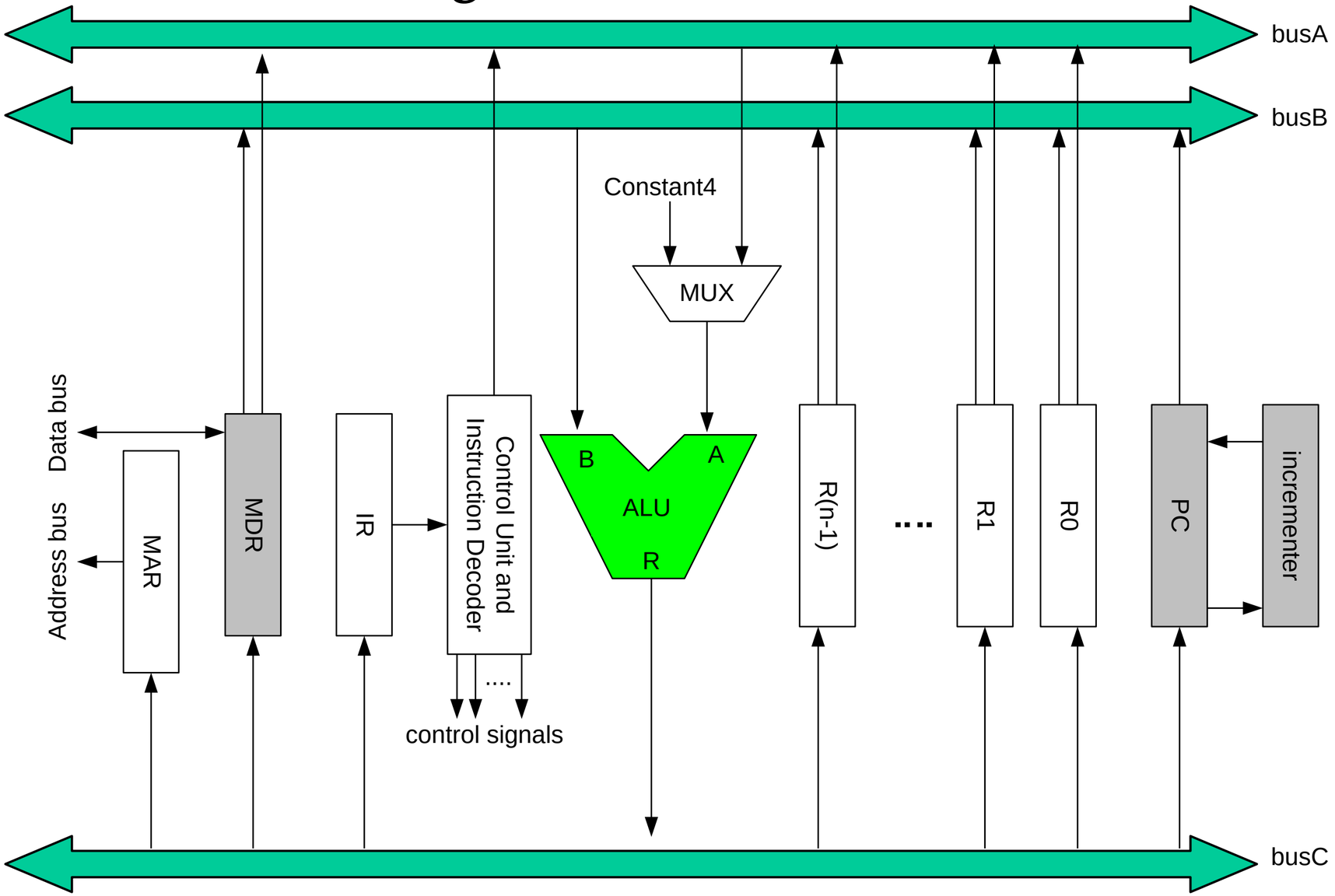
Multiple-Bus Organization

- Allow the contents of two different registers to be accessed simultaneously and have their contents placed on buses A and B.
- Allow the data on bus C to be loaded into a third register during the same clock cycle.
- Incrementer unit.
- ALU simply passes one of its two input operands unmodified to bus C

→control signal: R=A or R=B

Step	Action
1	PC_{out}, R=B, MAR_{in}, Read, incPC
2	WMFC, $MDR_{in\_from\_databus}$
3	MDR_{out_busB}, R= B, IR_{in}
4	$R6_{out\_busA,}R5_{out\_busB}$, Add, $R4_{inbusC}$, End

Three-bus organization



Instruction *ADD R6,R5, R4*

Step	Action
1	PC_{out}, R=B, MAR_{in}, Read, incPC
2	WMFC, $MDR_{in\_from\_databus}$
3	MDR_{out_busB}, R= B, IR_{in}
4	$R6_{out\_busA}$, $R5_{out\_busB}$, Add, $R4_{inbusC}$,
End	

Control Units

2 types of Control units
- Hardwired
- Microprogrammed

Hence, the required control signals are uniquely determined by the following information:

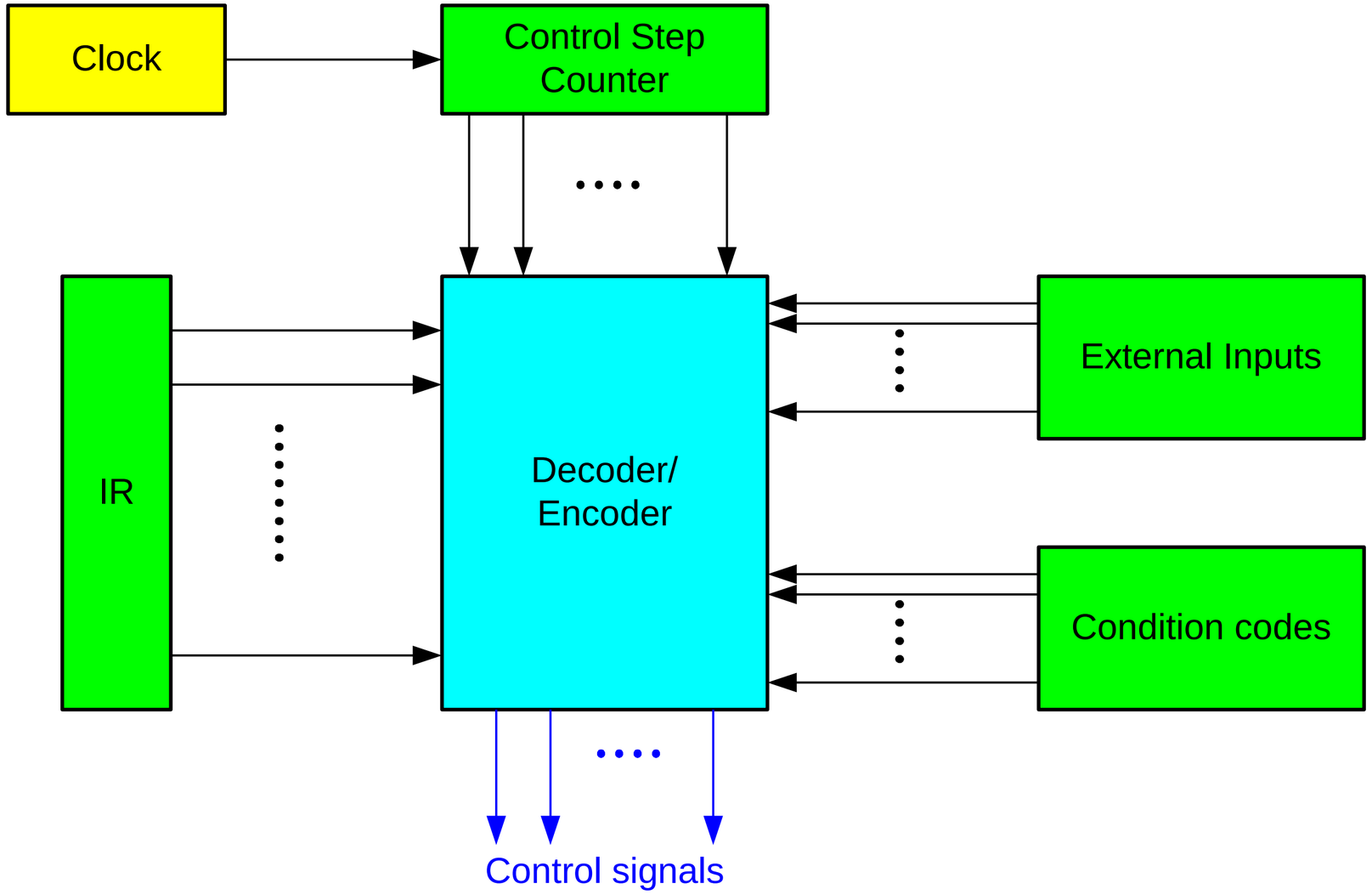
Contents of the control counter.

Contents of the instruction register.

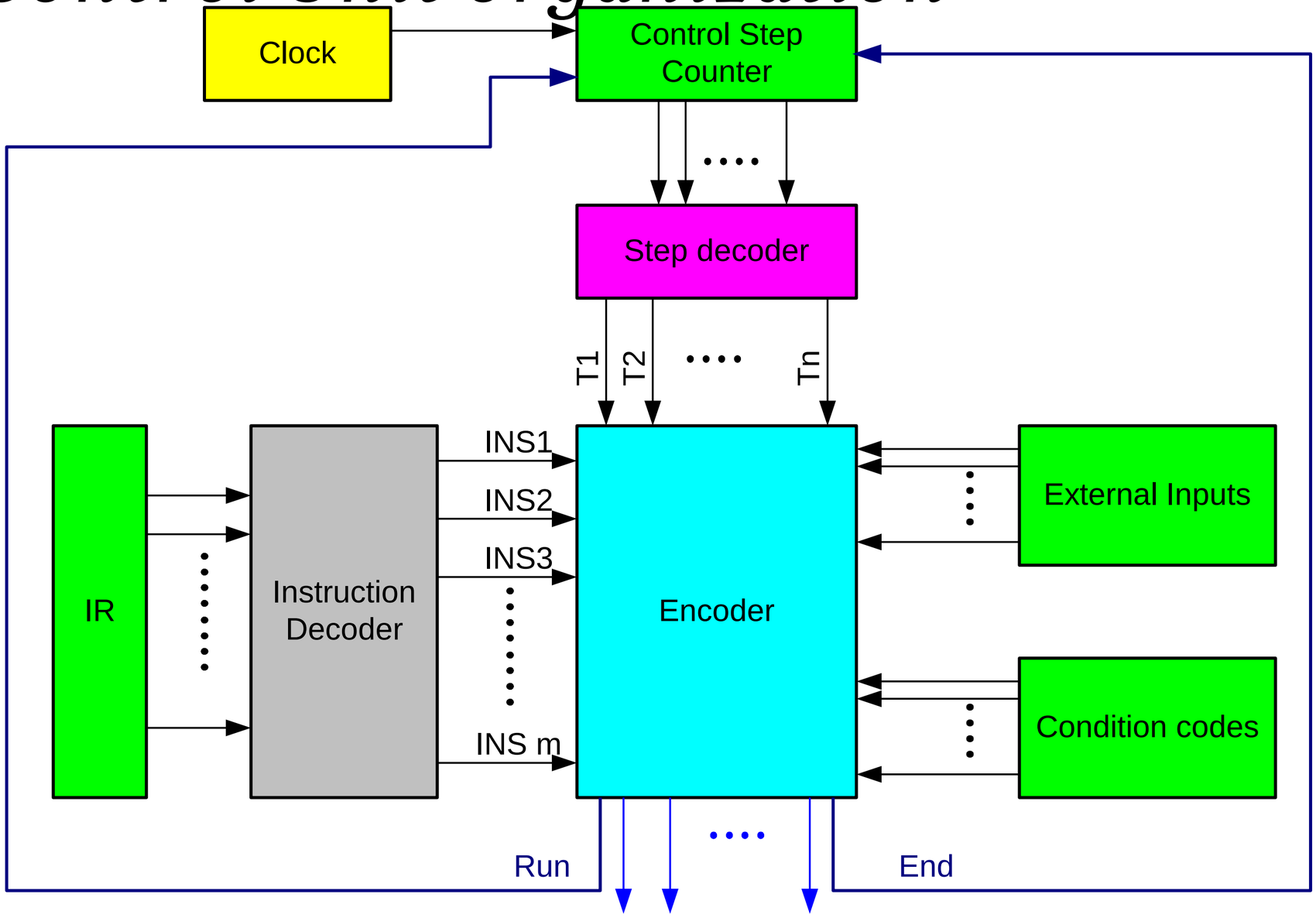
Contents of the condition code and other status flags.

The external inputs represent the state of the CPU and various control lines connected to it, such as *MFC* status signal. The condition codes/ status flags indicates the state of the CPU. These includes the status flags like carry, overflow, zero, etc.

Hardwired Control Unit

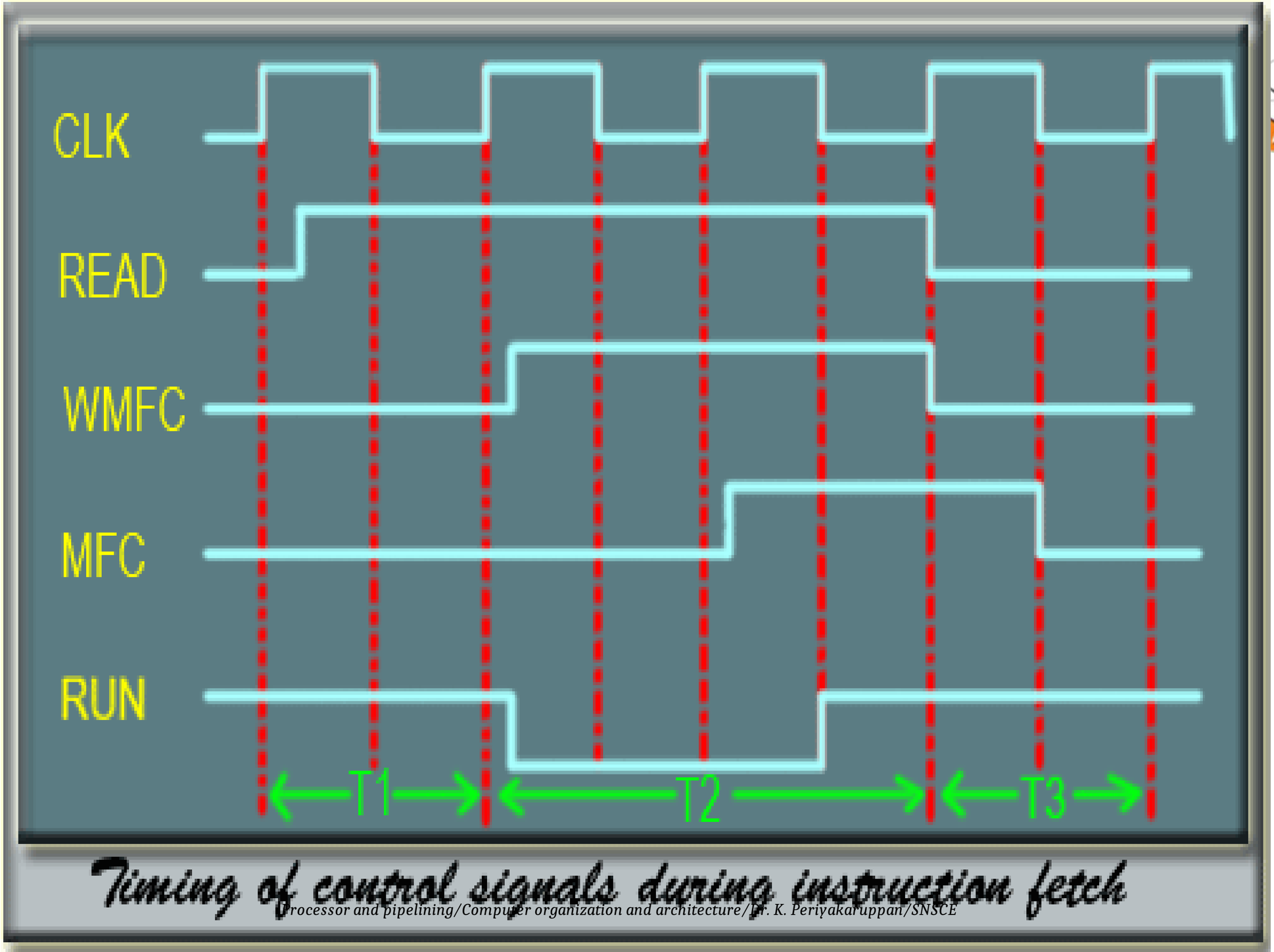


Control Unit organization



The decoder part of decoder/encoder part provide a separate signal line for each control step, or time slot in the control sequence.

Similarly, the output of the instructor decoder consists of a separate line for each machine instruction loaded in the IR, one of the output line *INS1* to *INSm* is set to 1 and all other lines are set to 0.



Timing of control signals during instruction fetch

Z_{in} and END control signals

Zin = T1 + (T6·ADD) + (T4 · BR)+…..

End = (T7 ·ADD) + (T5 ·BR) + (((T5 ·N)+(T4 ·N)) ·BRN)+....

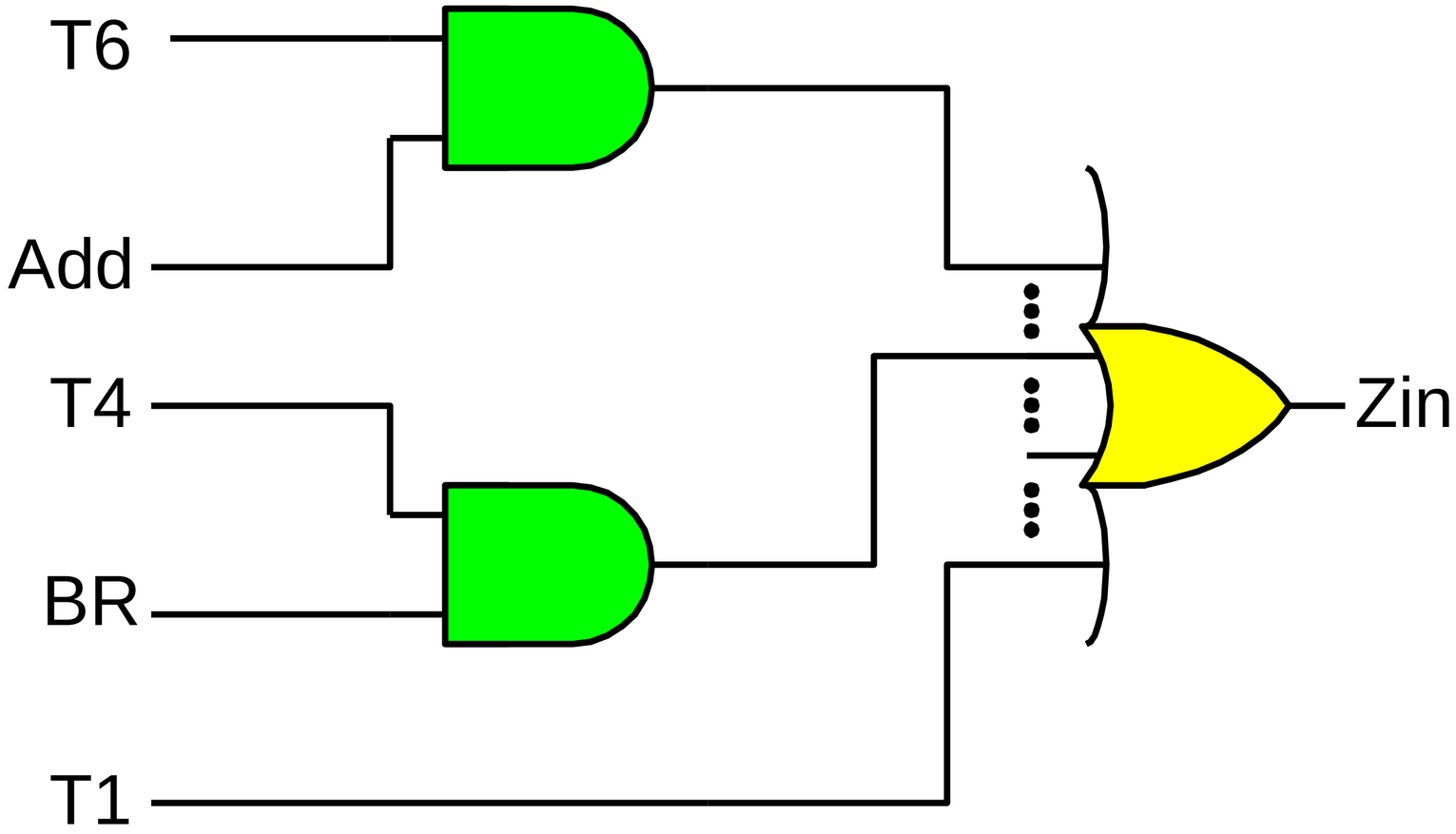
Note

BR = Branch instruction

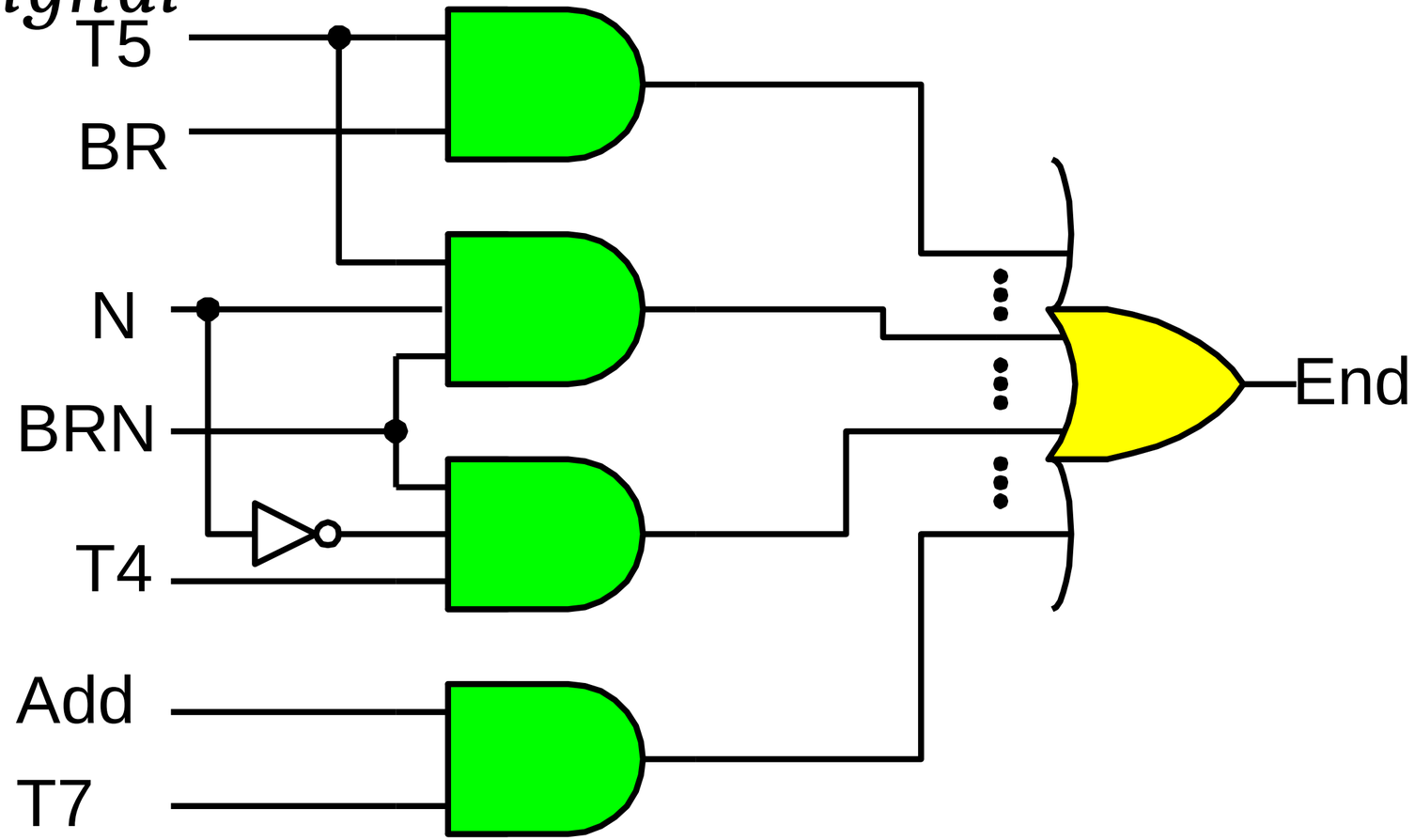
BRN = Branch<0 instruction

N = Negative flag

Generation of Z_{in} control signal



Generation of END control signal



Micro programmed control unit

In microprogrammed control unit, the logic of the control (control signals) unit is specified by a microprogram.

Terminologies

Control Word (CW) :

Control word is defined as a word whose individual bits represent the various control signal. Therefore each of the control steps in the control sequence of an instruction defines a unique combination of 0s and 1s in the CW.

A sequence of control words (CWs) corresponding to the control sequence of a machine instruction constitutes the **micro program** for that instruction.

The individual control words in this microprogram are referred to as **microinstructions.**

The microprograms corresponding to the instruction set of a computer are stored in a special memory which will be referred to as the **microprogram memory or control store**.

"Control words" stored in "Control Store"

Micro - instruction	..	PC_{in}	PC_{out}	MAR_{in}	Read	MDR_{out}	IR_{in}	Y_{in}	Select	Add	Z_{in}	Z_{out}	$R1_{out}$	$R3_{in}$	$R3_{out}$	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	

From Figure 7.15 page 430 of "Computer Organization", 5th edition, Carl Hamacher, McGraw Hill

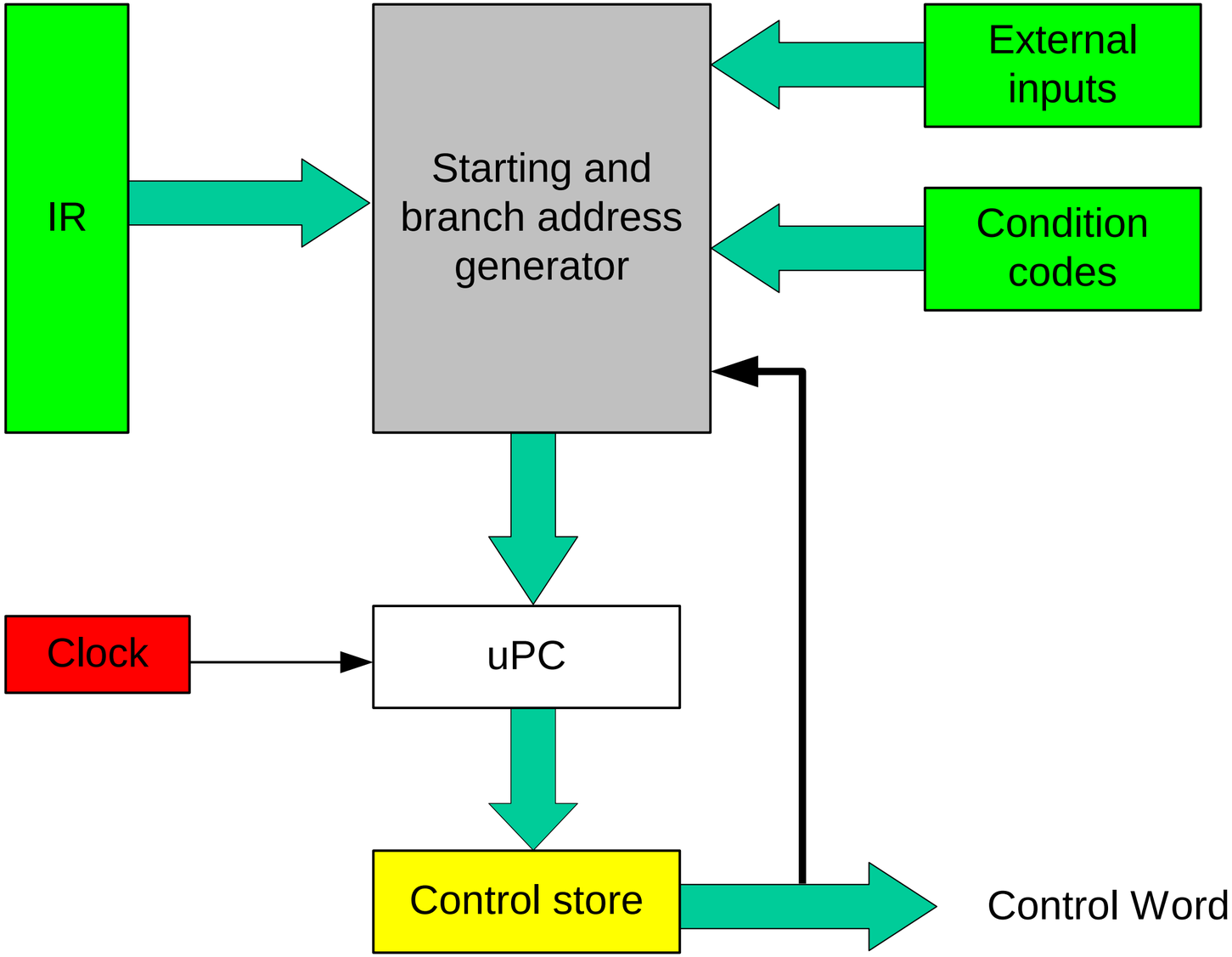
Microprogram consists of microinstructions which are nothing but the strings of 0's and 1's.

If the contents of the memory cell is 0, it indicates that the signal is not generated and if the contents of memory cell is 1, it indicates to generate that control signal .

The control unit can generate the control signals for any instruction by sequencially reading the CWs of the corresponding microprogram from the microprogram memory.
To read the control word sequentially from the microprogram memory a microprogram counter (PC) is needed.

The "starting address generator" block is responsible for loading the starting address of the microprogram into the PC every time a new instruction is loaded in the IR.
The PC is then automatically incremented by the clock, and it reads the successive microinstruction from memory.

Microprogrammed control unit



Microprogrammed Control

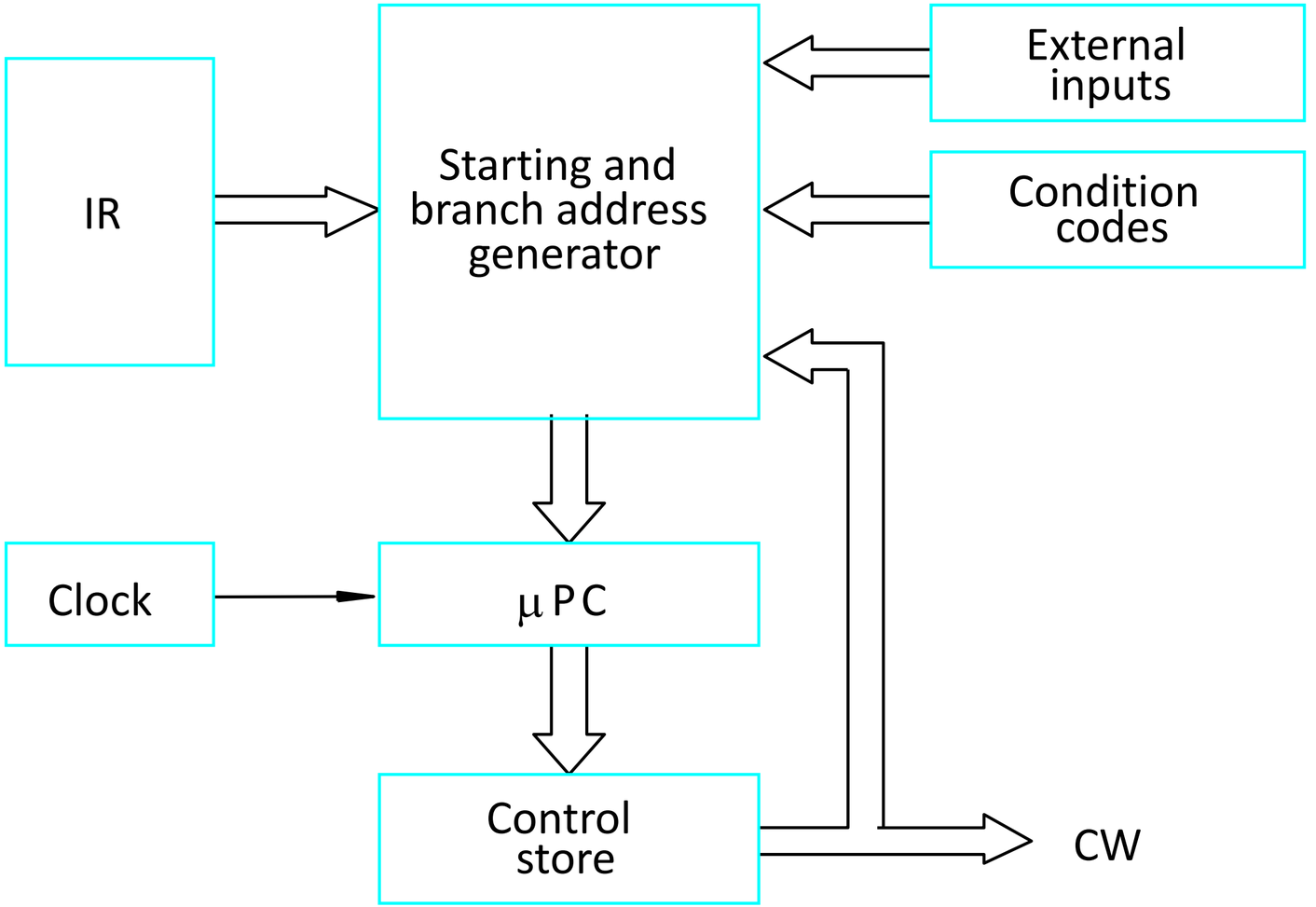


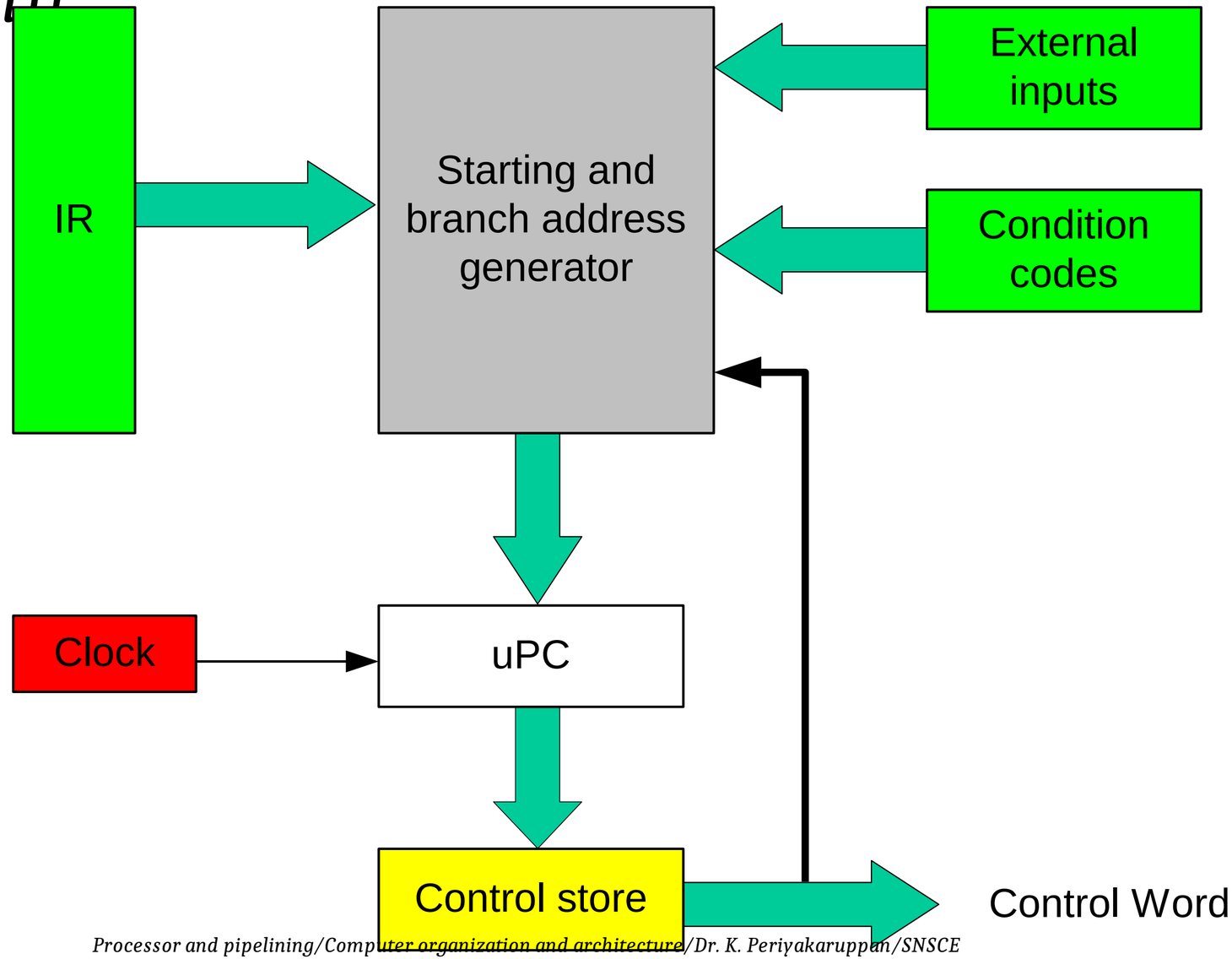
Figure 7.18. Organization of the control unit to allow conditional branching in the microprogram.

We have some instructions whose execution depends on the status of condition codes and status flag, as for example, the **branch instruction**. During branch instruction execution, it is required to take the decision between the alternative action.

To handle such type of instructions ,it is required to include some conditional branch microinstructions.

To support microprogram branching, the organization of control unit should be modified to accommodate the branching decision.

Microprogrammed control unit



The control bits of the microinstructions word which specify the branch conditions and address are fed to the "Starting and branch address generator" block.

This block performs the function of loading a new address into the PC when the condition of branch instruction is satisfied.

At the end of fetch microprogram, the starting address generator unit calculate the appropriate starting address of the microprogram for the instruction which is currently present in IR.

During the execution of a microprogram, the PC is always incremented everytime, a new microinstruction is fetched from the microprogram memory, except in the following situations:

1.When an End instruction is encountered, the PC is loaded with the address of the first CW in the microprogram for the instruction fetch cycle.

2.When a new instruction is loaded into the IR, the PC is loaded with the starting address of the microprogram for that instruction.

3.When a branch microinstruction is encountered, and the branch condition is satisfied, the PC is loaded with the branch address.

Microprogrammed Control

ADD
NUM,R1

Steps	Actions
1.	PCout, MARin, Read, select4, Add, Zin
2.	Zout, PCin, Wait For MFC
3.	MDRout, IRin
4.	Address-field-of-IRout, MARin, Read
5.	R1out, Yin, Wait for MFC
6.	MDRout, Add, Zin
7.	Zout, R1in
8.	END

Microprogrammed Control

Control sequence for Conditional Branch instruction (BRN) Branch on negative)

Steps	Actions
1.	PCout, MARin, Read, Select4 , Add , Zin
2.	Zout, PCin, Wait for MFC
3.	MDRout, IRin
4.	PCout, Yin
5.	Address field-of IRout,SelectY, Add, Zin
6.	Zout, PCin
7.	End

Assessment

What is

1.Single bus organization____

2. Multiple bus organization___

3.Hard wired control unit ____

4.Microprogrammed control unit_____

Reference

1. Carl Hamacher, Zvonko Vranesic and Safwat Zaky, “Computer Organization”, McGraw-Hill, 6th Edition 2012.