Control Unit Implementation

The control unit of the CPU is responsible for sending all the hardware signals required both by internal and external units of the CPU for proper execution of instructions.

There are two popular methods available to design the CU.

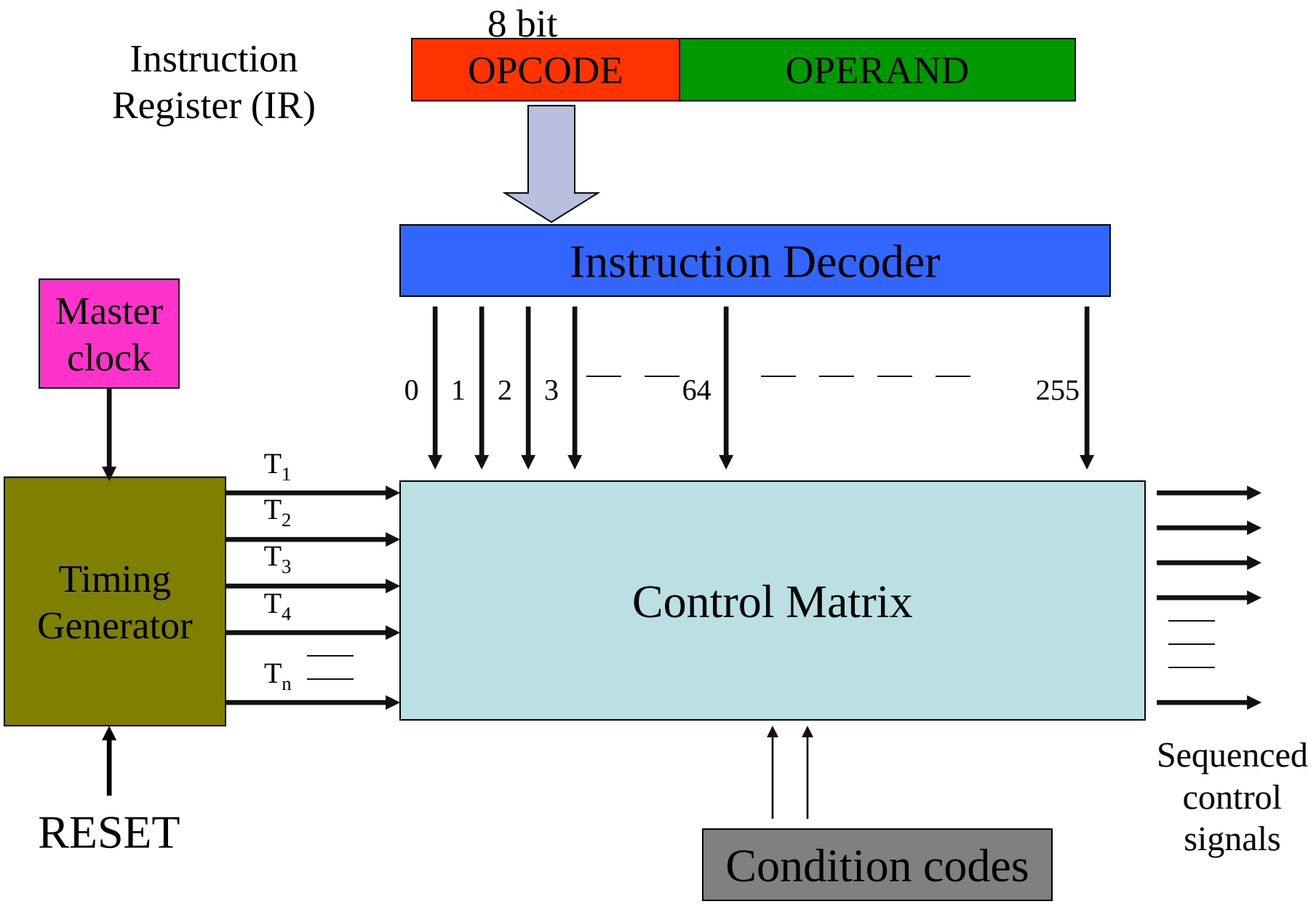
- Hard wired control unit (Hardware approach)
- Micro program control unit (software approach)

The hardwired approach has the

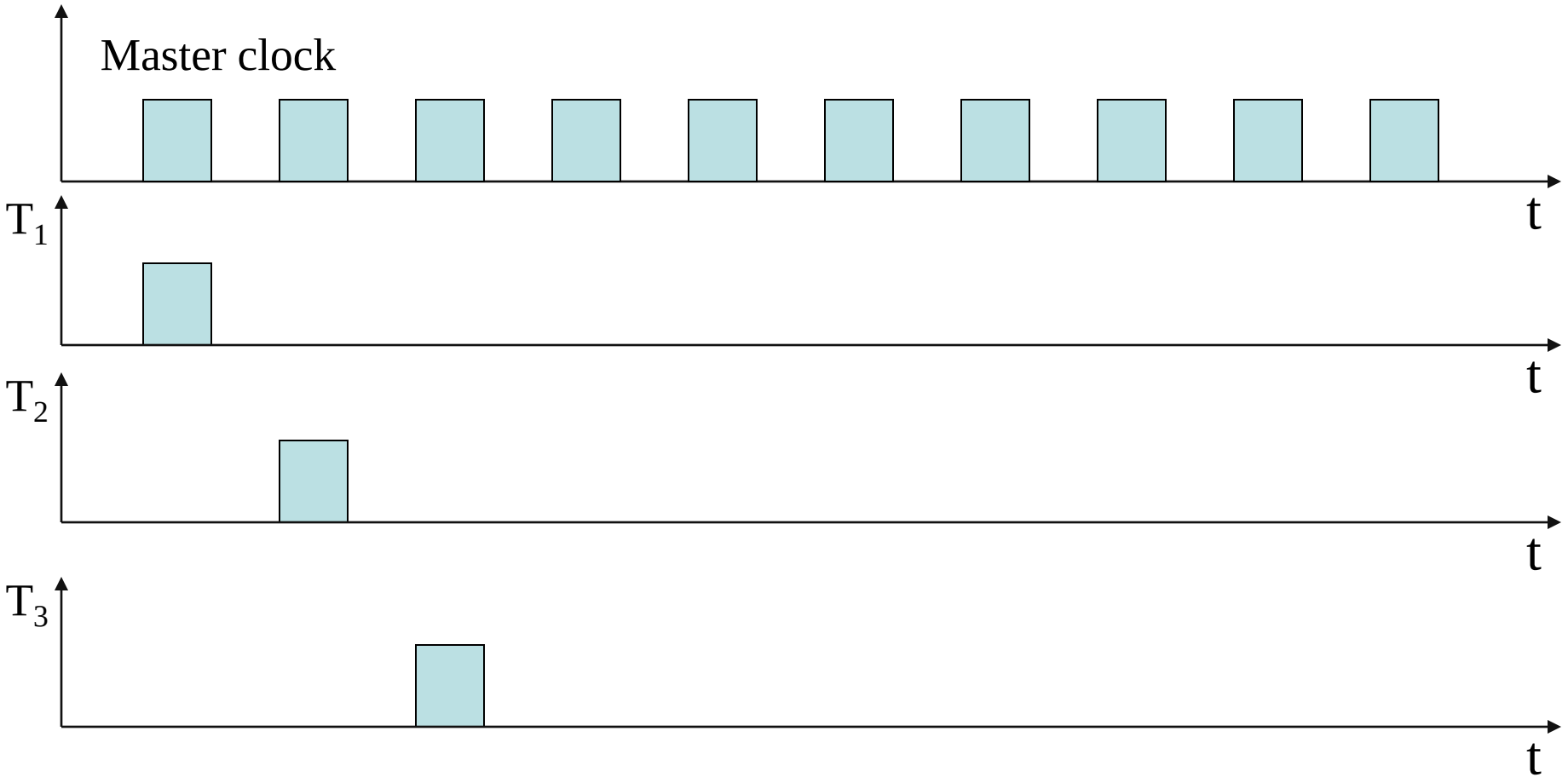
advantage that it is possible to devise faster CPUs if typical CPU speeds are faster than typical memory speeds (a situation that has been true for quite some time).

drawback of hardwired logic is that it is difficult to design CPUs with large and complex instruction sets using this approach.

Hard wired control unit



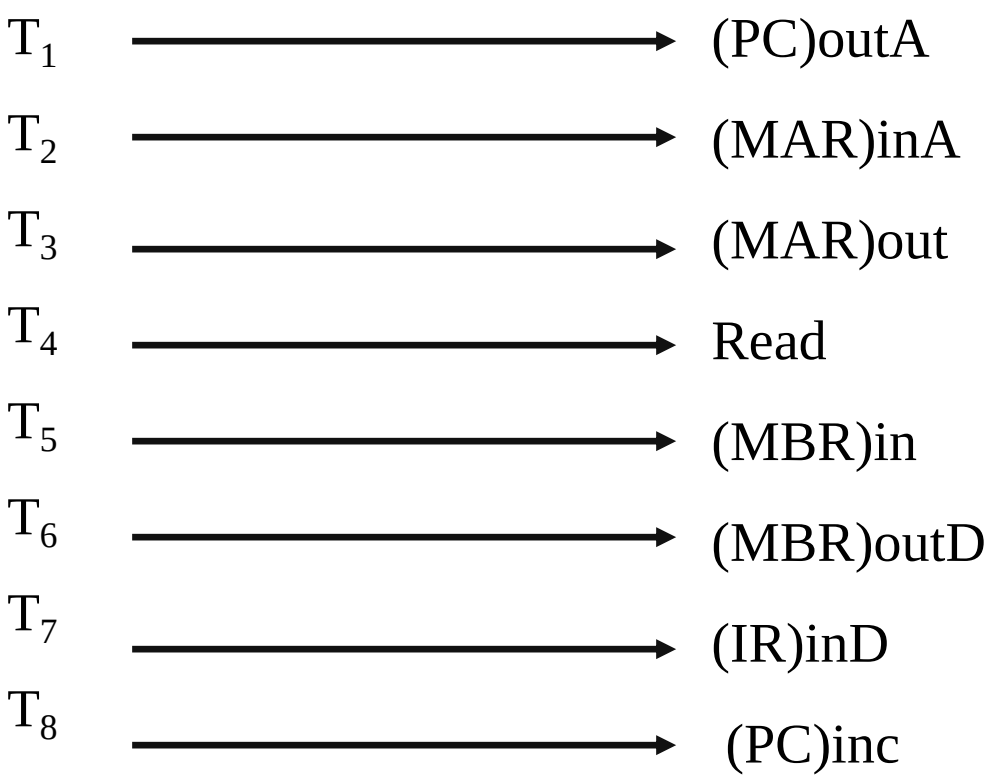
The timing generator generates pulses at corresponding time slots as shown below



Similarly all the timing signals are generated up to T_n. Then start again from T_1. Same thing repeats in a cyclic manner.

Now consider the fetch cycle.

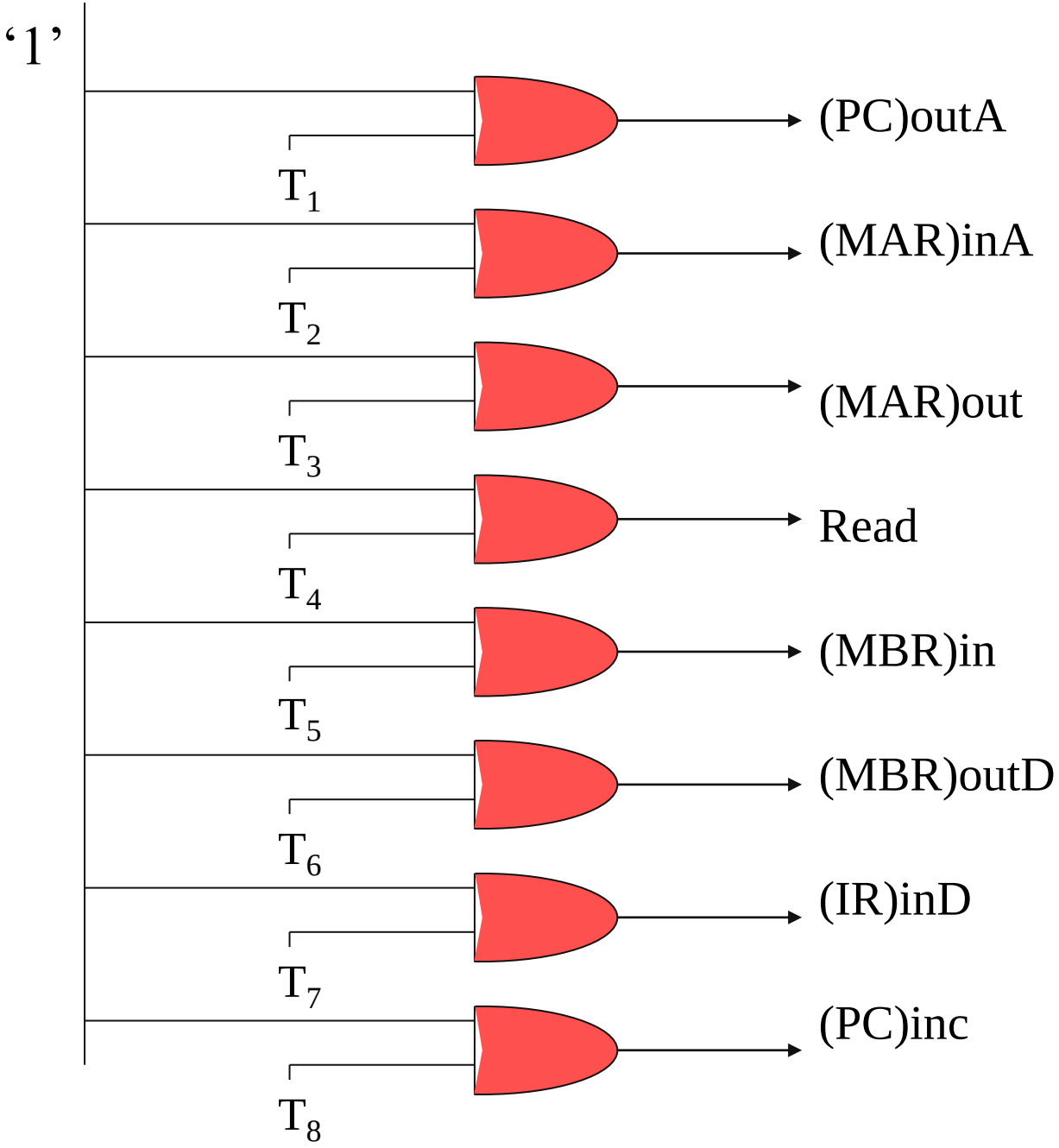
Control signals to activate are listed below according to the order



The above control signals are activated in the given order.

For that timing generator outputs are hard wired to an array of AND gates.

Selected Line corresponding to the OPcode -64 line LOAD A



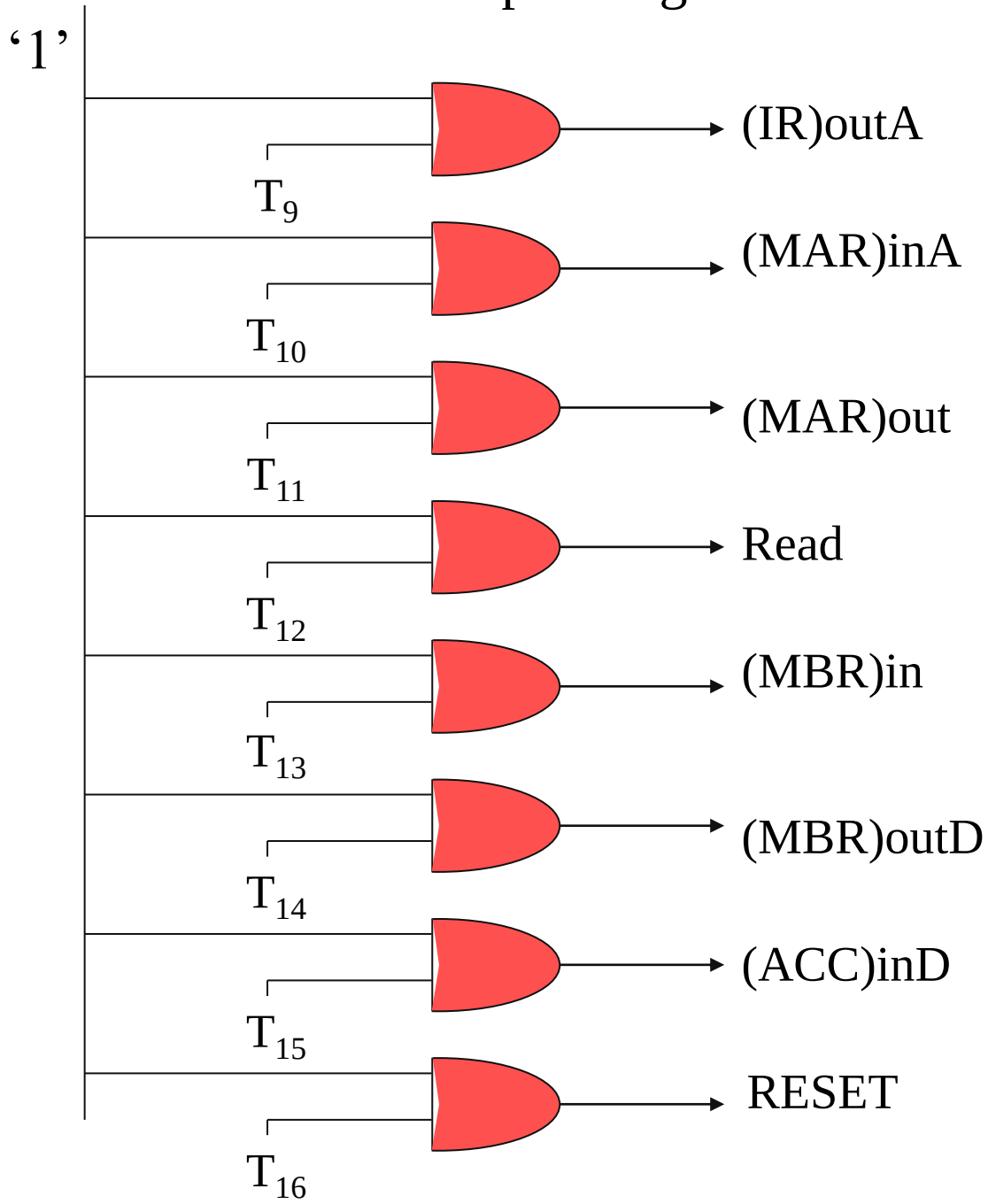
- Instruction decoder select a line out of 255 according to the Opcode part (64th line for LOAD A OPcode).

- Each and every line is connected to a set of AND gates.

- Required control signals are activated at the proper order as the timing pulses are applied to gate inputs
- This continues for the execution cycle as well.

- Now consider the execution cycle of LOAD A instruction

Selected Line corresponding to the OPcode -64 line LOAD A



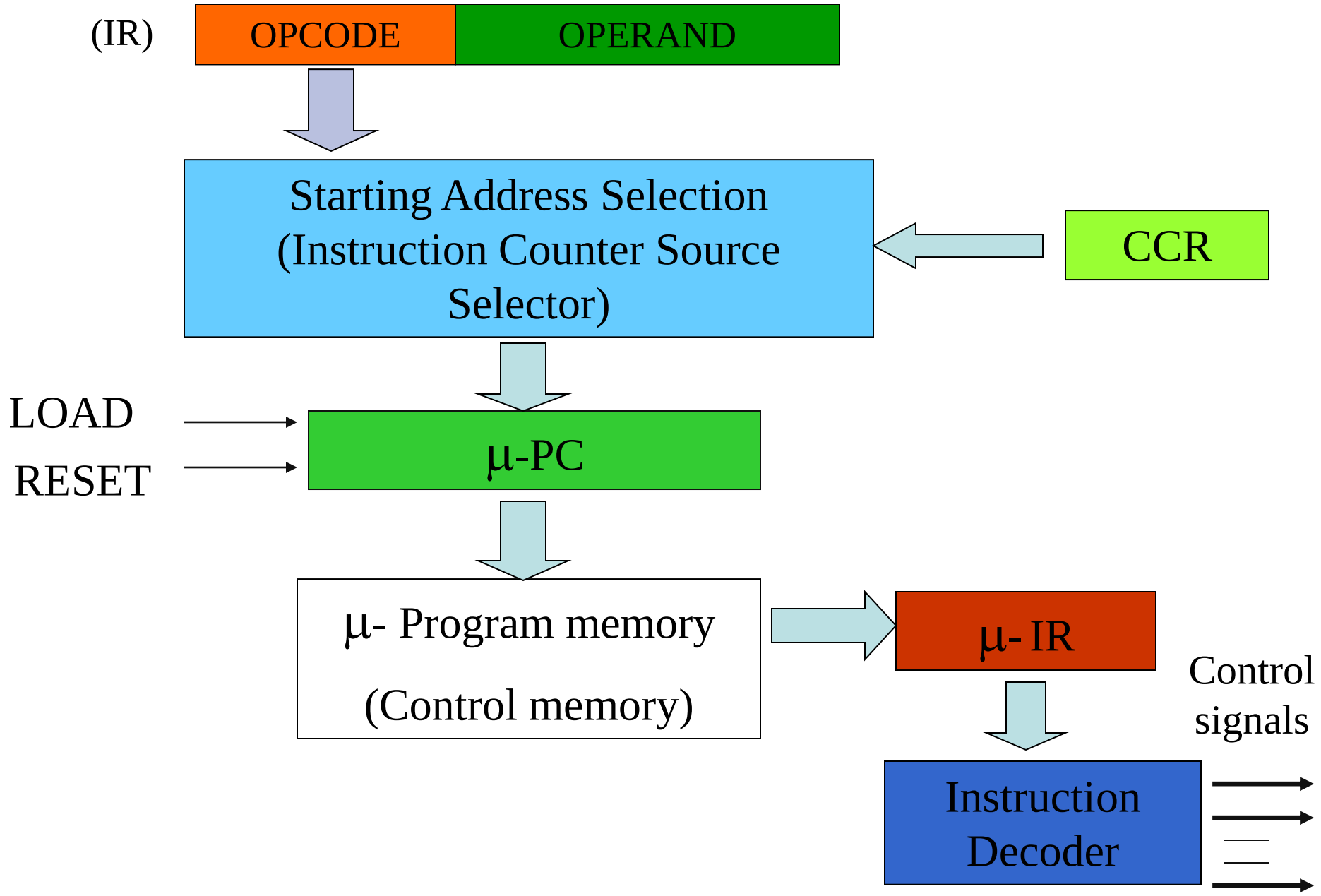
- The RESET control signal initializes the timing generator.
- In the above execution cycle, the timing generator is reset with T16 (with 16th clock pulse).
- Therefore with the following clock pulse the timing generator will activate T1 there by starting the fetch cycle of the next instruction.
- Since the length of the execution cycle can vary from instruction to instruction timing generator will be initilized during different time slots(not always with T16)

Draw the AND gate band diagram for the FETCH & EXECUTION cycles of the ADD A,(address) instruction

Draw the AND gate band diagram for the FETCH & EXECUTION cycles of the STORE A,(address) instruction

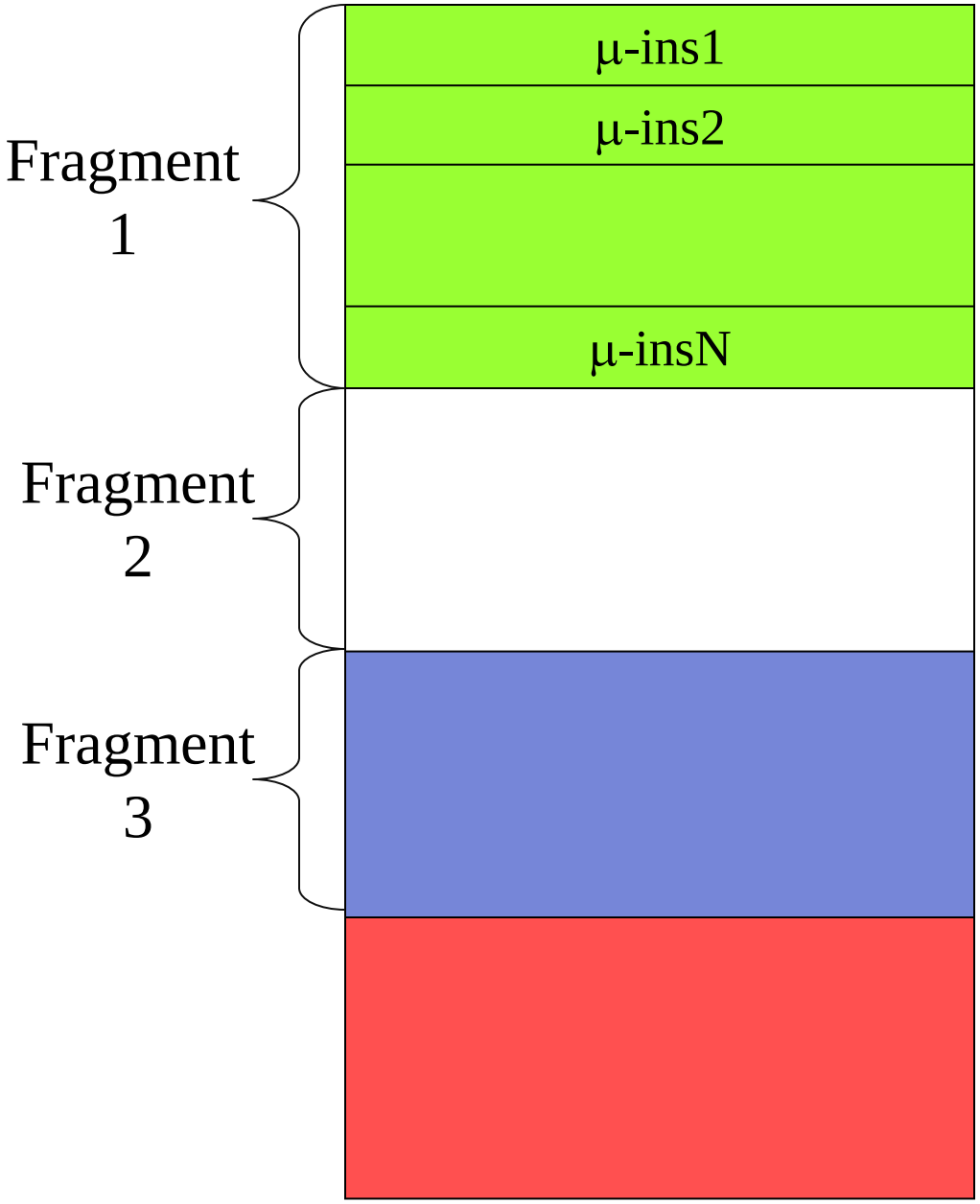
Draw the AND gate band diagram for the FETCH & EXECUTION cycles of the JUMP (address) instruction

Micro Program Control Unit



The control memory is divided in to number of micro instruction segments. Each segment contains number of micro instructions.

There is a micro instruction fragment corresponding to fetch cycle. usually the first one, fragnent1. Similarly there is a micro instruction fragment corresponding for each execution cycle , of the machine code instruction cycle.



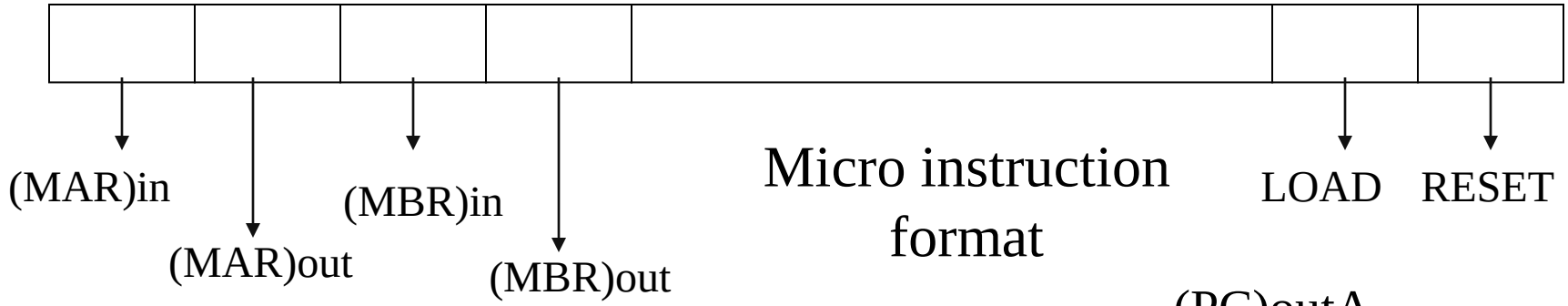
The instruction counter source selector (ICSS) provides the starting address of the micro instruction fragment corresponding with the current machine code instruction.

The micro program counter starts counting from this starting address.

The micro instruction currently being executed, is held in the micro instruction register,and is decoded by the instruction decoder for final control outputs.

- At the end of each fetch cycle micro instruction program counter is loaded by the output of the ICSS at the end of each execution cycle the μ-PC is reset, initiating the next fetch cycle.
- Micro program CU can be implemented in three different ways.
    1. Horizontal microprogramming
    2. Vertical microprogramming
    3. Combined microprogramming

Horizontal microprogramming



In horizontal micro programming a unique bit position in the micro instruction format is reserved for each control

Each micro instruction is responsible for the activation of a single control signal.

Let us consider the fetch cycle. The control sequence for the fetch cycle is given here

(PC)outA

(MAR)inA

(MAR)out

Read

(MBR)in

(MBR)outD

(IR)inD

(PC)inc

Clock cycle	(MAR)in A	(MAR)out	(MBR)in D	(MAR)in	(MBR)out D	(MBR)out	READ	WRITE	(PC)inc	(PC)out A	(IR)in D	(MAR)in A		LOAD	RESET
1	0	0	0	0	0	0	0	0	0	1	0			0	0
2	1	0	0	0	0	0	0	0	0	0	0			0	0
3	0	1	0	0	0	0	0	0	0	0	0			0	0
4	0	0	0	0	0	0	1	0	0	0	0			0	0
5	0	0	0	1	0	0	0	0	0	0	0			0	0
6	0	0	0	0	1	0	0	0	0	0	0			0	0
7	0	0	0	0	0	0	0	0	0	0	1			0	0
8	0	0	0	0	0	0	0	0	1	0	0			0	0
9	0	0	0	0	0	0	0	0	0	0	0			1	0

In horizontal micro programming an instruction decoder is not necessary inside the control unit. Control signals are straight away derived from the micro instruction register.

Each out put of micro instruction register correspond to a control signal output.

In horizontal microprogramming the length of the micro instruction is relatively large.(If there are 200 control signals to be generated, each micro instruction will be 200 bit long). This is considered to be a disadvantage of the horizontal microprogramming.

However this technique supports the parallel operation of controls. If parallel operation of controls are provided a micro instruction can contain more than a single one ‘1’.

Further advantage of this technique is that it does not require an instruction decoder

Draw the control memory for the execution cycle of ADD A,(address) instruction

Draw the control memory for the execution cycle of STORE A,(address) instruction

Vertical microprogramming

This technique is an effort to reduce the micro instruction length. There by reducing the complexity of micro IR and the control memory.

Ex assume that there are 256 controls 7 each micro instruction activates a single control out of the 256 controls.

Since $256 = 2^8$ an 8 bit coded micro instruction can be used this technique is known as Vertical micro programming and an instruction decoder is required to get the final control outputs.

The advantages of vertical micro programming are the reduced micro instruction length & control memory size.

The disadvantages are Parallel operation of controls is not possible. Additional instruction decoder is required. Due to the additional instruction decoder the speed of the control unit can be slower compared to horizontal micro programming.

Combined microprogramming

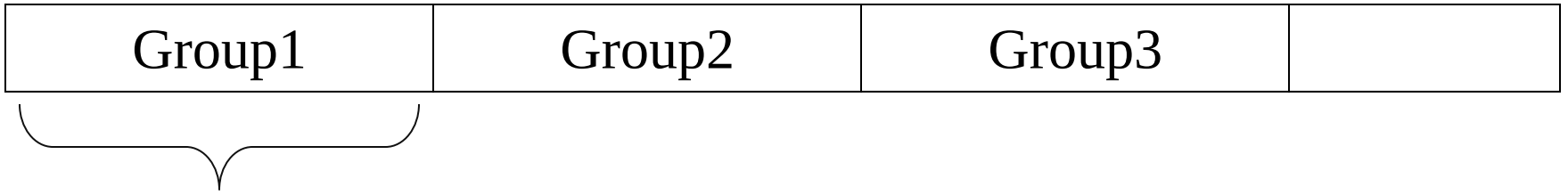
This is a hybrid version of horizontal and vertical micro programming. This technique tries to reduce the micro instruction length, while allowing parallel operation of controls.

In this technique the control signals are grouped in such a way so that parallel operation of controls is not required within a group.

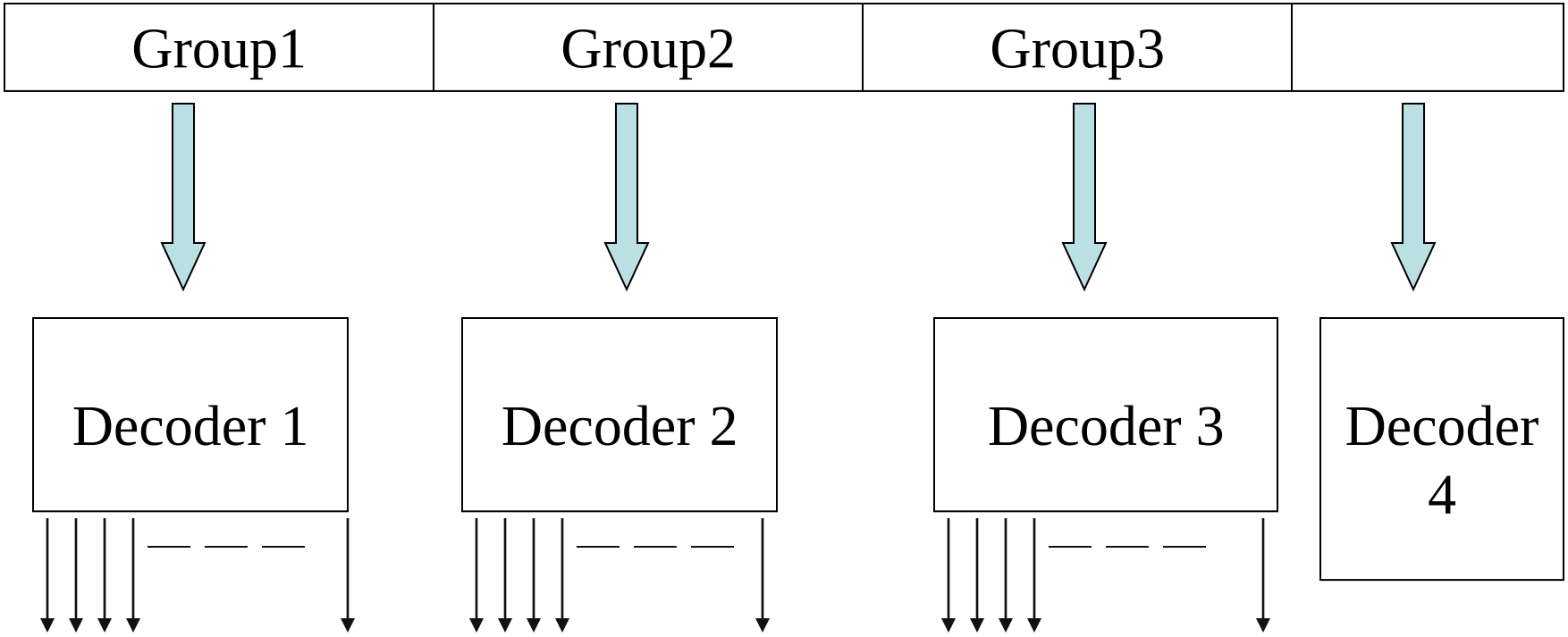
e.g. group1 Bus D controls (ACC)out D (R_0)out D (TEMP)out D)…

group2 Bus D out put controls (ACC)in D (R_0)in D …

group3 ALU controls.



Each group require separate decoder unit.



Within each group a fully encoded bit pattern can be used so that only a single control within a group can be activated at a time.

However control signals from different groups can be activated at the same time.

An additional bit pattern within each group is required to keep all the control signals disabled, each other pattern would corresponds to the activation of a single control within the group.