ARCOS Group

uc3m | Universidad **Carlos III** de Madrid

Lesson 4 (II)
The processor

Computer Structure

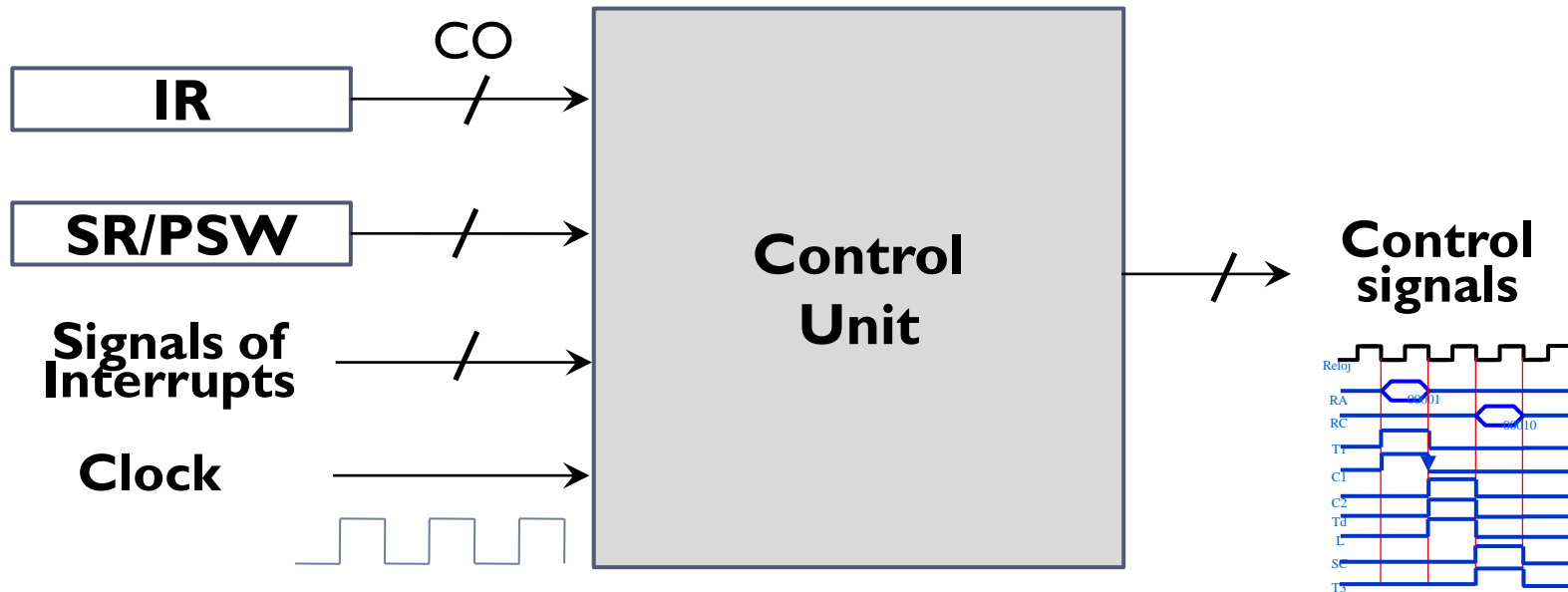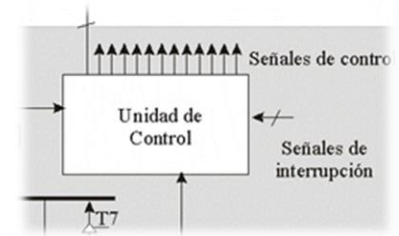Bachelor in Computer Science and Engineering

Contents

1. Computer elements
2. Processor organization
3. Control unit
4. Execution of instructions
5. Control unit design
6. Execution modes
7. Interrupts
8. Computer startup
9. Performance and parallelism

Control Unit

IR — CO —/→

SR/PSW —/→

Signals of Interrupts —/→

Clock —→

→ **Control Unit** →/→ **Control signals**
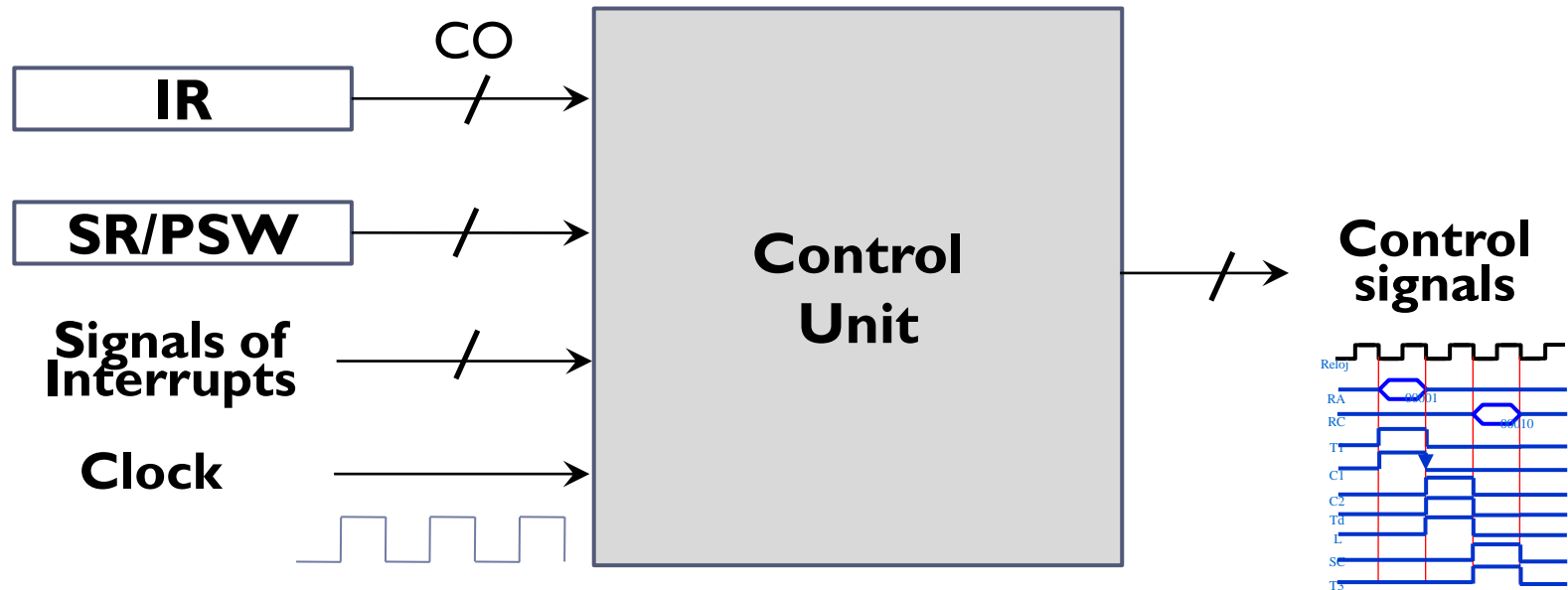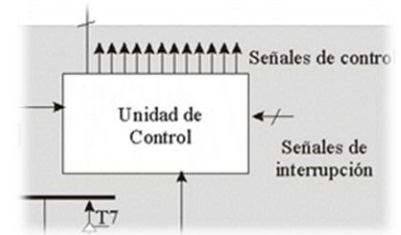
Control Unit

- Every **control signal** is **function** of the values **of**:
    - The content of the **IR**
    - The content of **SR**
    - The **period of time (clock)**

Félix García-Carballeira, Alejandro Calderón Mateos

Control unit design

▸ For each machine instruction:

1. Define the behavior using RTL (register transfer language) for every clock cycle.

2. Translate the behavior to values of each control signal at each clock cycle

3. Design a circuit that generates the value of each control signal at each clock cycle

Control unit design

▸ For each machine instruction:

1. Define the behavior using RTL (register transfer language) for every clock cycle.

2. Translate the behavior to values of each control signal at each clock cycle

3. Design a circuit that generates the value of each control signal at each clock cycle

Control unit design

Instruction   *mv R0 R1*

Sequence of
**elementary
operations**
- RI <- [PC]
- PC++
- decoding
- *R0 <- R1*

Sequence of **control
signals** for each
elementary operation

Reloj
FD
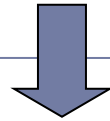CM
L
E
TM
FEM

Ciclo de
Lectura

Control unit design

▶ For each machine instruction:

1. Define the behavior using RTL (register transfer language) for every clock cycle.

2. Translate the behavior to values of each control signal at each clock cycle

3. Design a circuit that generates the value of each control signal at each clock cycle

Control techniques
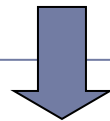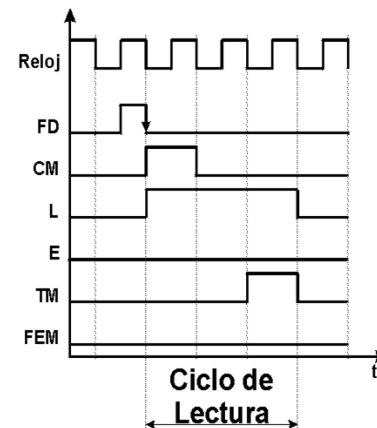
▸ ## Hardwired (relay logic) control unit
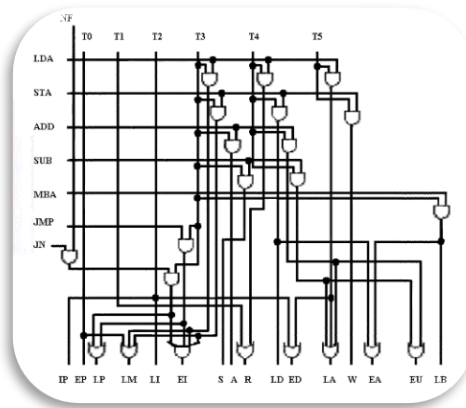
▸ ## Microprogrammed control unit
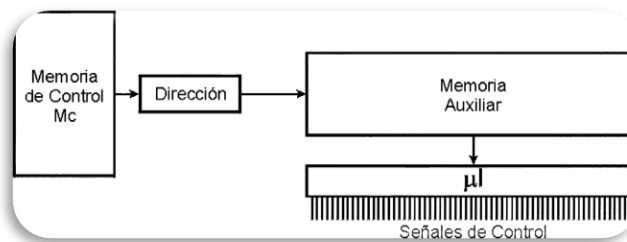
Example

▸ Design of a control unit for a set of 4 machine instructions.

▸ Instructions to consider:
  ▸ `add Rd, Rf:      Rd <- Rd + Rf`
  ▸ `lw Rd, dir:      Rd <- MP[dir]`
  ▸ `sw Rf, dir:      MP[dir] <- Rf`
  ▸ `bz R, dir:       if (R==0) PC<- dir`

State machine for the example

Example for a computer with only 4 machine instructions

```
add rd, rf
lw rd, dir
sw Rf, dir
bz R, dir
```

ARCOS @ UC3M
Félix García-Carballeira, Alejandro Calderón Mateos
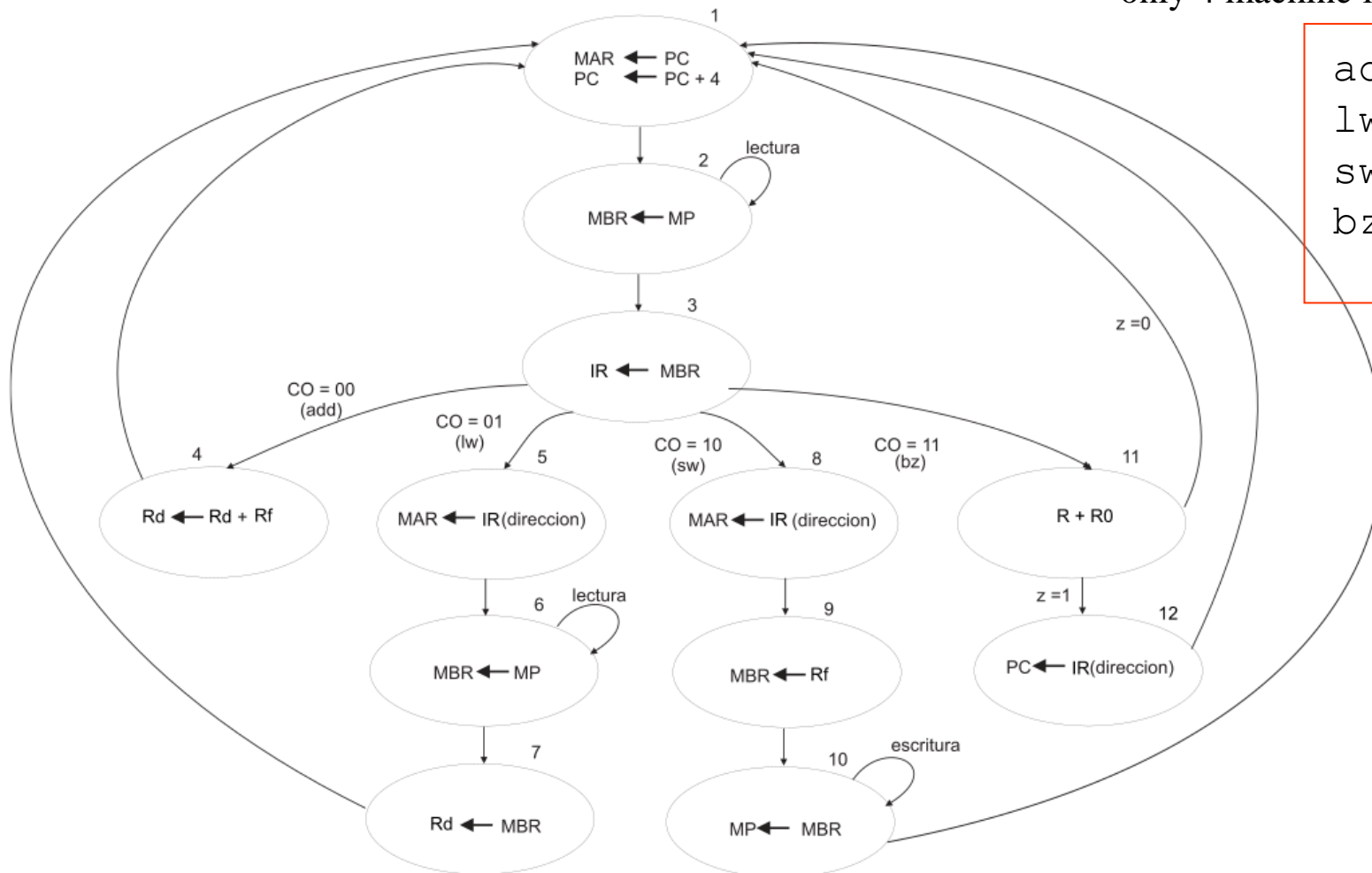
Control techniques

▸ **Two techniques** to design and build the control unit:
   a) **Relay logic**
   b) **Programmable logic (microprogrammed)**

ARCOS @ UC3M
Félix García-Carballeira, Alejandro Calderón Mateos
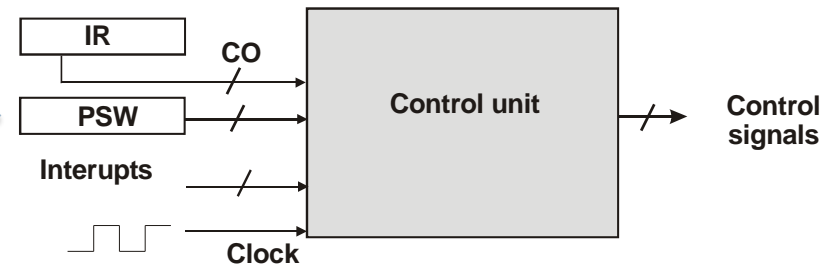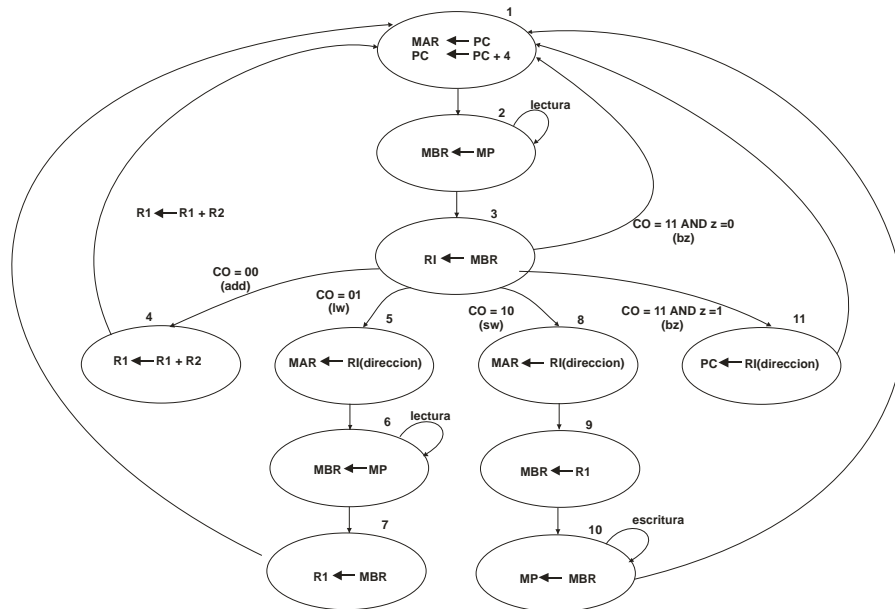
Control Unit: relay logic

▸ Construction by means of logic gates, following logic design methods.

▸ Characteristics:

  ▸ Laborious and costly circuit design and tuning.

  ▸ Difficult to modify:

    ▸ Complete redesign.

  ▸ Very fast (used in RISC computers).
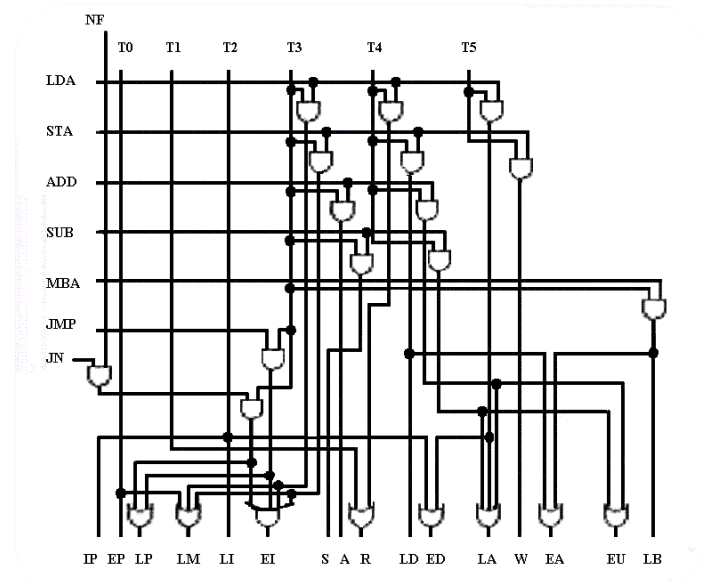
Control Unit: programmable logic
microprogramming

▶ Basic idea: Use a memory (control store) to store the signals of each cycle of each instruction..

▶ Characteristics:

  ▶ Easy modification

    ▶ Upgrade, expansion, etc.

    ▶ E.g.: Certain consoles, routers, etc.

  ▶ Easy to have complex instructions

    ▶ E.g.: Diagnostic routines, etc.

  ▶ Easy to have several sets of instructions

    ▶ Other computers can be emulated.

  ▶ Simple HW $\Rightarrow$ hard microcode
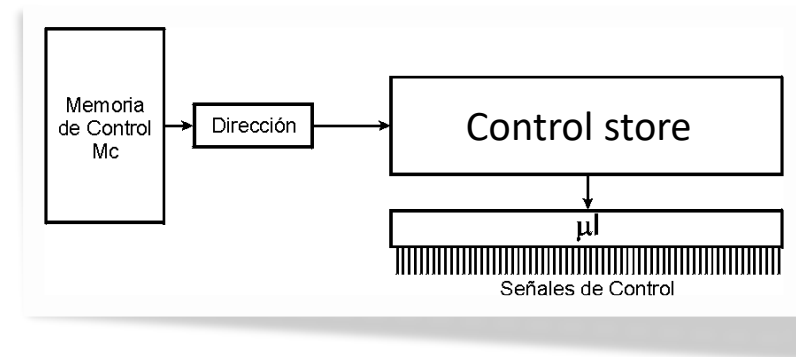
General structure of a microprogrammed control unit
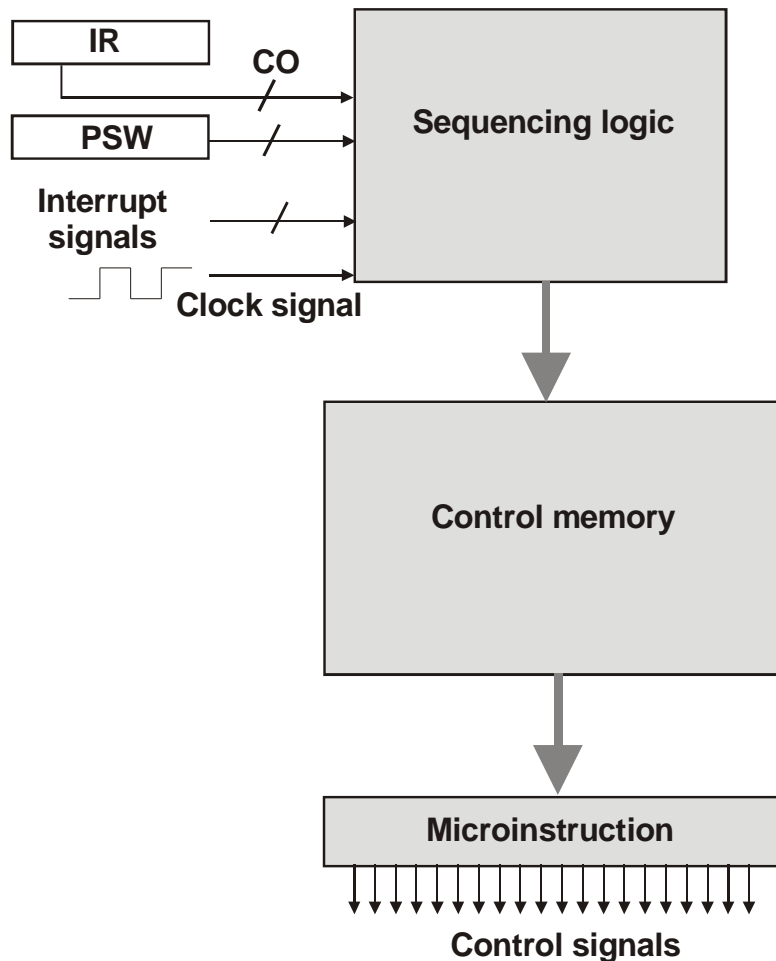
C1	C2	C3	C4	R5	C6	C7	C8	C9	C10	C11	Td	Ta	T1	T2	T3	T4	T5	T6	T7	T8	RA4	RA3	RA2	Ra1
0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0

RB4	Rb3	RB2	RB1	RB0	RC4	RC3	RC2	RC1	RC0	SC	L	E	Cop3	Cop2	Cop1	Cop0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

ARCOS @ UC3M
Félix García-Carballeira, Alejandro Calderón Mateos

Microprogrammed control unit.
Microinstructions

	TP	FD	CM	L	E	TM	FI	DAO	DA1	DA2	DA3	DB0	DB1	DB2	DB3	XY1	XY2	XX1	XX2	OP0	OP1	OP2	OP3	TA	CR	TD	FP	FEM	FLM	FEST	PO
I_1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
I_2	1	1	0	0	0	0	0	0	1	0	1	1	1	0	0	1	1	1	1	1	0	1	1	1	0	0	0	0	0	1	1
I_3	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
I_4	0	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
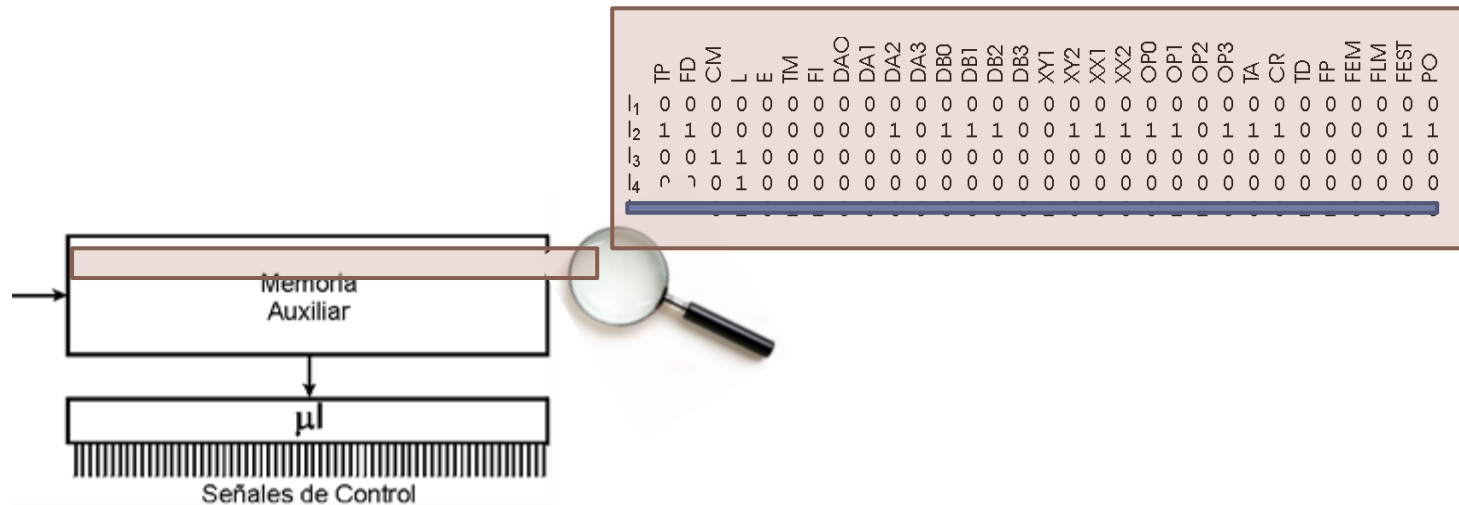
▸ **Microinstruction:** To each word defining the value of each control signal in a cycle of an instruction/fetch+IAC

▸ The **microinstructions**

  ▸ have one bit for each control signal.

  ▸ A list of 1's and 0's representing the state of each control signal during a period of one instruction.

Microprogrammed control unit.
microprogram and microcode
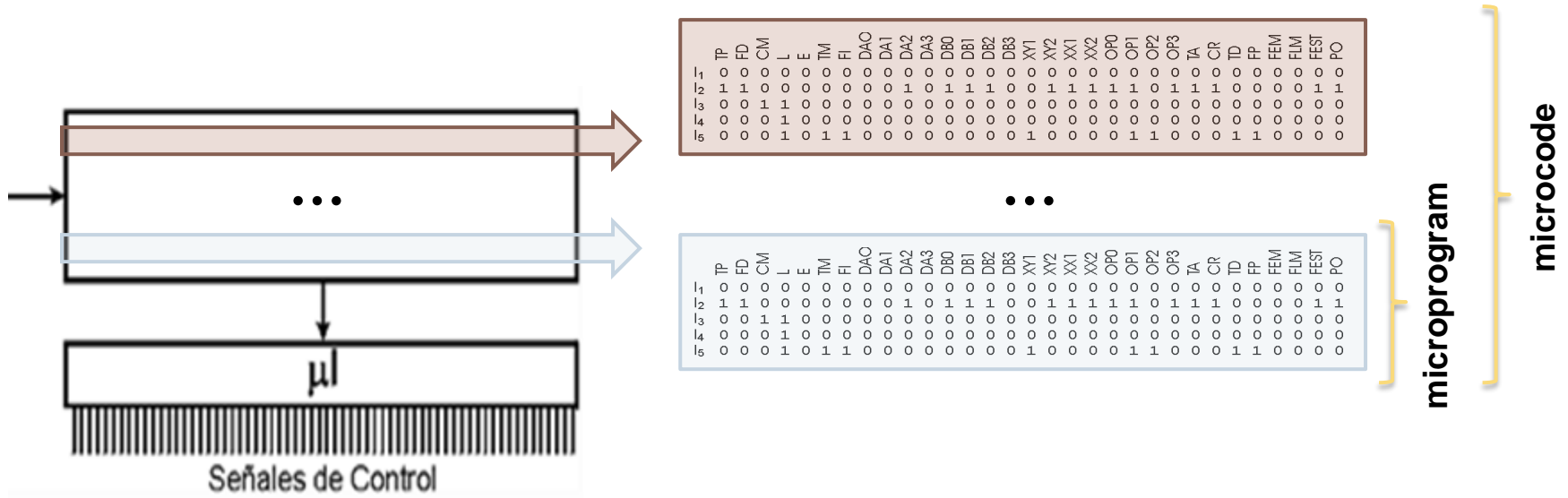
- **microprogram**: ordered list of microinstructions, which represent the chronogram of a machine instruction.
- **microcode**: set of microprograms of a machine.

Contents of the control memory

FETCH

li $a0, #value

…

j dir

- ‣ FETCH: get next instruction
  - ‣ IAC: interrupt acknowledge cycle.
  - ‣ IR<- Mem[PC], PC++, jump-to-O.C.

- ‣ Microprograms:
  one for every machine instruction
  - ‣ fetch rest of operands (if any)
    - ‣ Updates PC on multi-word instructions
  - ‣ Execute the instruction
  - ‣ Jump to FETCH

ARCOS @ UC3M
Félix García-Carballeira, Alejandro Calderón Mateos

Microprogrammed control unit structure

▶ **Three basic conditions:**

1. **Sufficient control memory to store all microprograms** corresponding to all instructions.

2. **Procedure for associating each instruction with its microprogram**
   ▶ Procedure that converts the instruction operation code to the control memory address where your microprogram starts..

3. **Sequencing mechanism** to **read successive microinstructions, and** to **branch to another microprogram** when the current one is finished.

▶ **Two alternatives:**

1. Explicit sequencing.
2. Implicit sequencing.

Microprogrammed C.U. structure with explicit sequencing
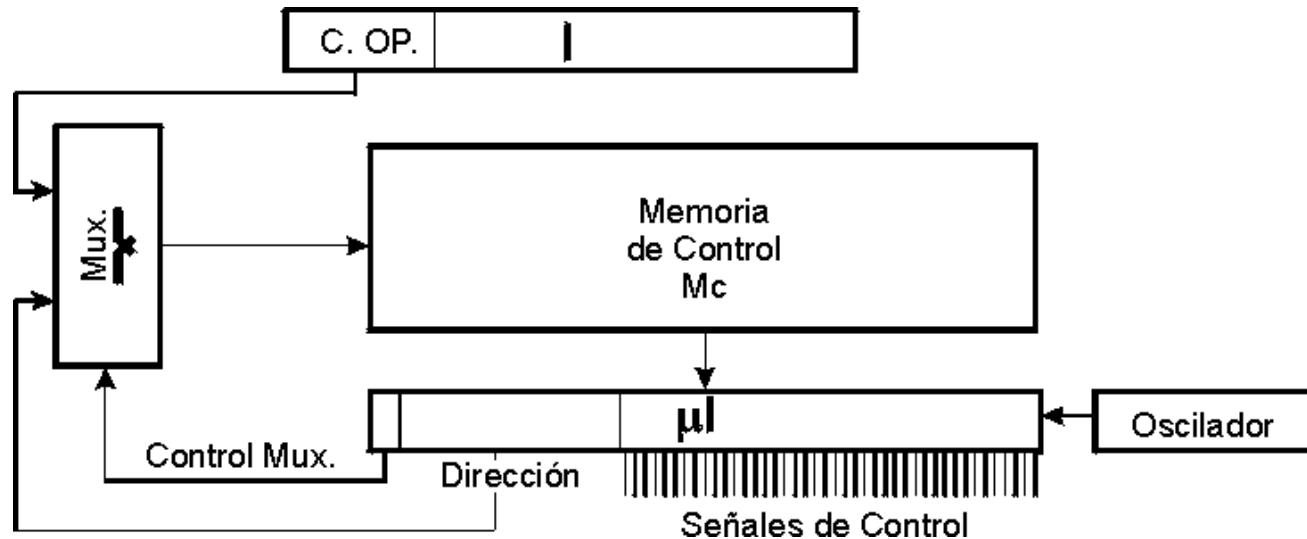
- ▸ Control memory stores all μprograms, where each μinstruction provides the next μinstruction μaddress
- ▸ The OC represents the μAddress of the first μinstruction associated with the machine instruction.
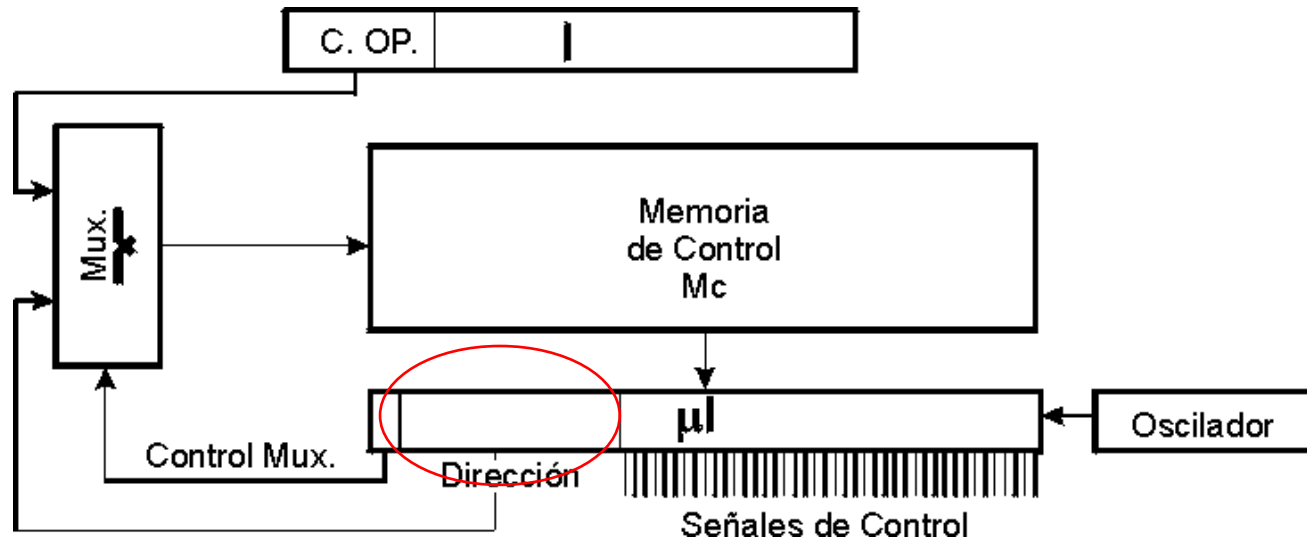
Microprogrammed C.U. structure with explicit sequencing

- Control memory stores all μprograms, where each μinstruction provides the next μinstruction μaddress
- Problem: large amount of control memory for instruction sequencing, required stores the next μaddress

Microprogrammed C.U. structure with implicit sequencing
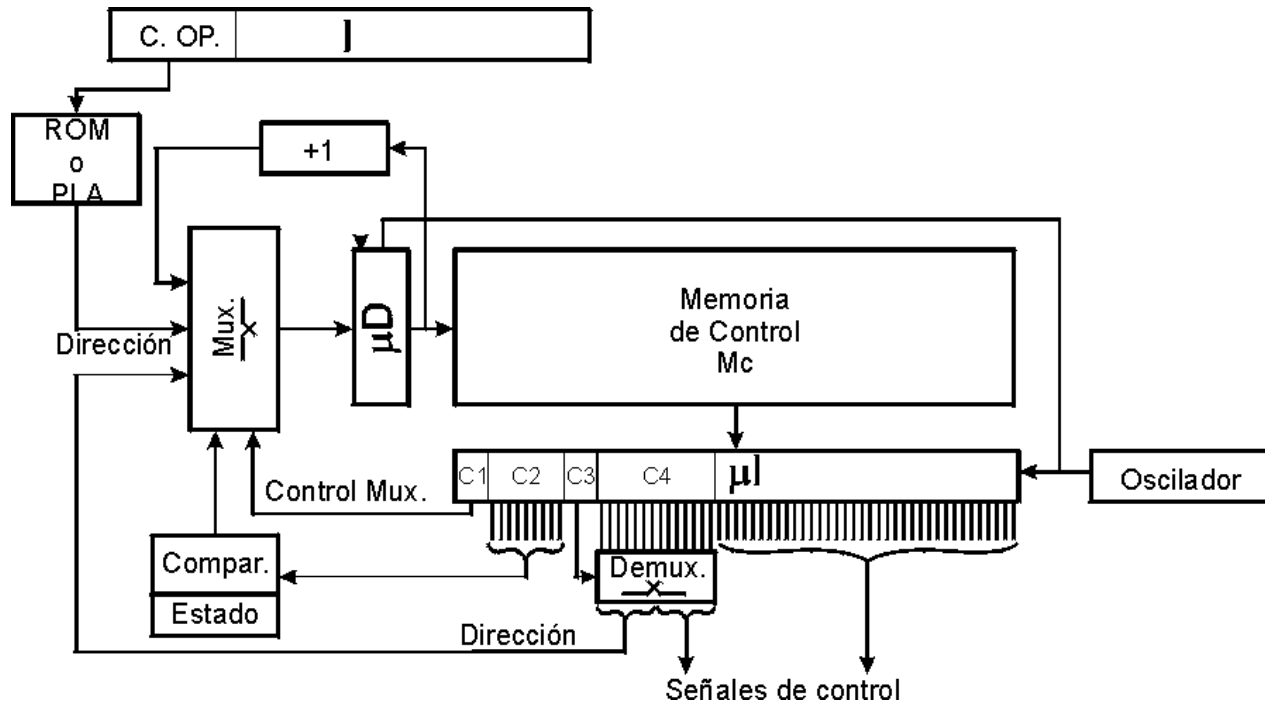
- ‣ Control memory stores all microprograms consecutively in the control memory.
- ‣ The ROM/PLA associates each instruction with its microprogram (first μaddress, μconditional μinstruction (+1), μconditional μbifurcations or μloops).
- ‣ Next μinstruction (+1), conditional μbifurcations or μloops

ARCOS @ UC3M
Félix García-Carballeira, Alejandro Calderón Mateos

Example of Control Unit with implicit sequencing

Control Memory

ROM (co2µAddr)

0	µAddr for 1st µI of µP 0
1	µAddr for 1st µI of µP 1
N	µAddr for 1st µI of µP N

Addr = OC

Control Memory	
µProgram for fetch	
µProgram for Inst. 0	µP 0
µProgram for Inst. 1	µP 1
µProgram for Inst. N	µP N

ARCOS @ UC3M
Félix García-Carballeira, Alejandro Calderón Mateos

Example of Control Unit with implicit sequencing

Control Memory

ROM (co2µAddr)

OC

0	µAddr for 1st µI of µP 0
1	µAddr for 1st µI of µP 1
N	µAddr for 1st µI of µP N

fetch

µI0
µI1
µIK (jump to µProg)

Example of Control Unit with implicit sequencing

Control Memory

ROM (co2µAddr)

OC

0	µAddr for 1st µI of µP 0
1	µAddr for 1st µI of µP 1
N	µAddr for 1st µI of µP N

fetch

µI0
µI1
µIK (jump to µProg)

Example of Control Unit with implicit sequencing

Control Memory

ROM (co2µAddr)

fetch

	µI0
	µI1
	µIK (jump to µProg)

OC

0	µAddr for 1st µI of µP 0
1	µAddr for 1st µI of µP 1
N	µAddr for 1st µI of µP N

The Operation Code (OC) is at the Instruction Register (IR)

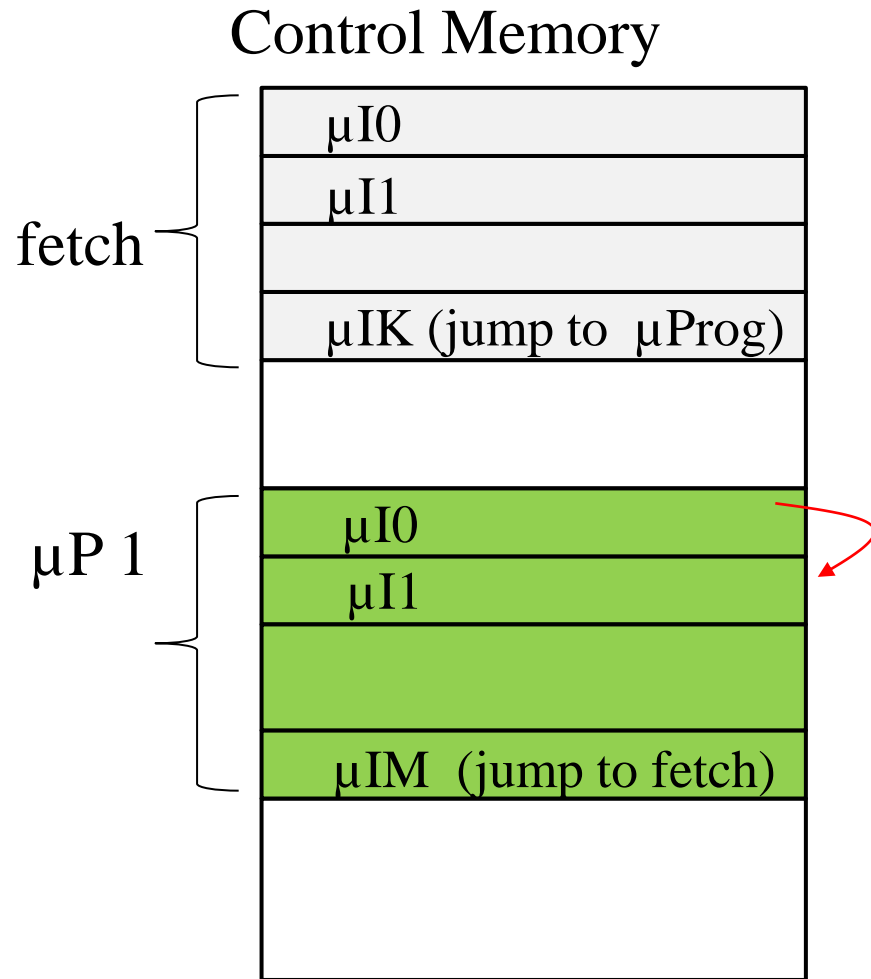# Example of Control Unit with implicit sequencing
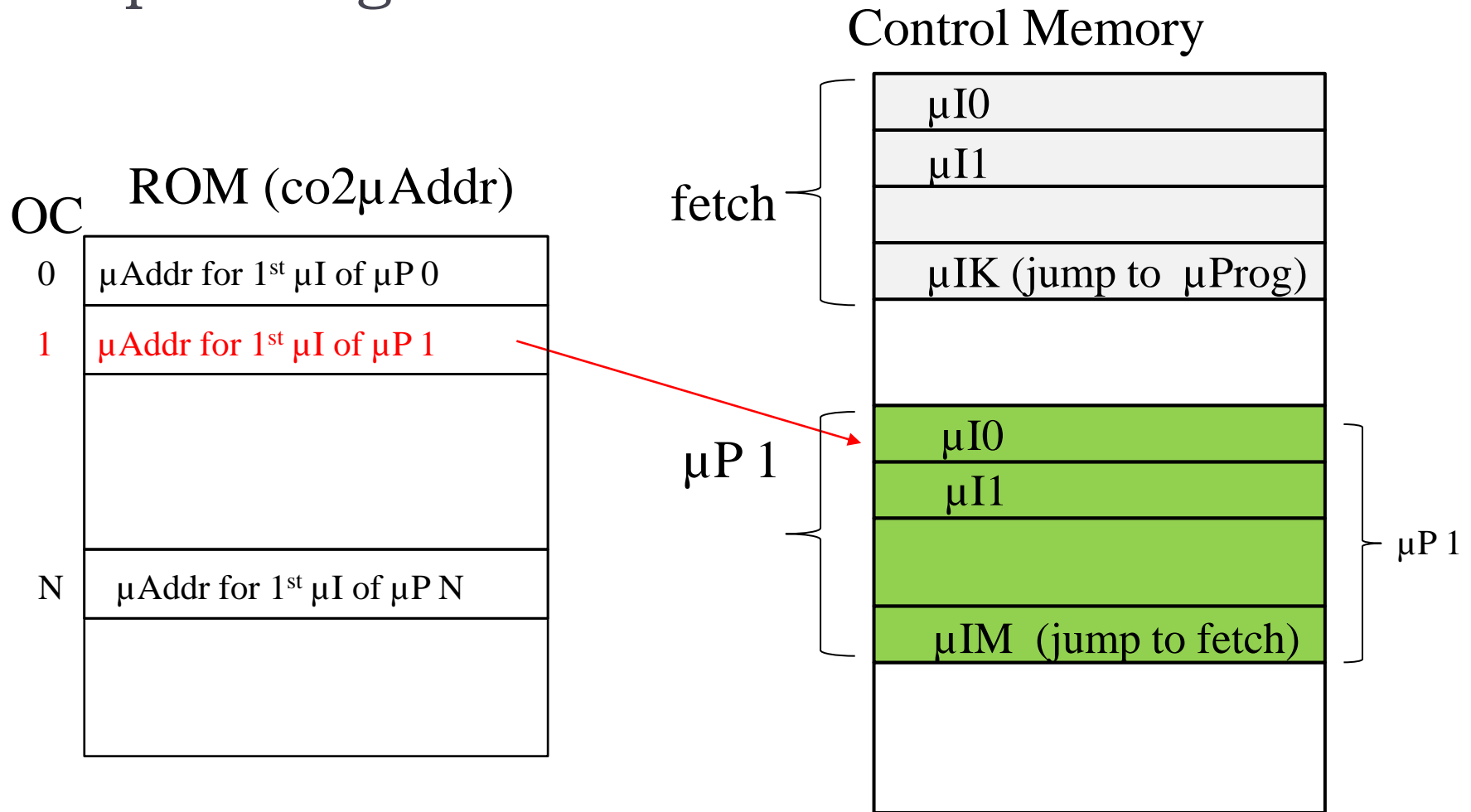
Control Memory

ROM (co2µAddr)

OC

0	µAddr for 1st µI of µP 0
1	µAddr for 1st µI of µP 1
N	µAddr for 1st µI of µP N

fetch
- µI0
- µI1
-
- µIK (jump to µProg)

µP 1

- µI0
- µI1
-
- µIM (jump to fetch)

µP 1

Example of Control Unit with implicit sequencing

Control Memory

ROM (co2μAddr)

OC

0	μAddr for 1st μI of μP 0
1	μAddr for 1st μI of μP 1
N	μAddr for 1st μI of μP N

fetch
μI0
μI1
μIK (jump to μProg)

μP 1
μI0
μI1
μIM (jump to fetch)

Example of Control Unit with implicit sequencing

Control Memory

μI0
μI1
μIK (jump to μProg)

fetch

ROM (co2μAddr)

OC

0	μAddr for 1st μI of μP 0
1	μAddr for 1st μI of μP 1
N	μAddr for 1st μI of μP N

μP 1

μI0
μI1
μIM (jump to fetch)

Example of Control Unit with implicit sequencing

Control Memory

ROM (co2µAddr)

OC

0	µAddr for 1st µI of µP 0
1	µAddr for 1st µI of µP 1
N	µAddr for 1st µI of µP N

fetch

µI0
µI1
µIK (jump to µProg)

µP 1

µI0
µI1
µIM (jump to fetch)

Félix García-Carballeira, Alejandro Calderón Mateos
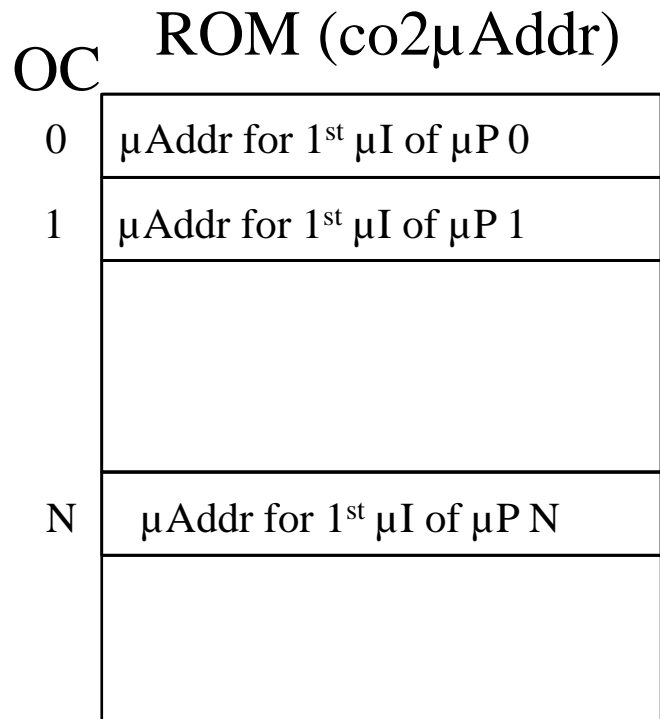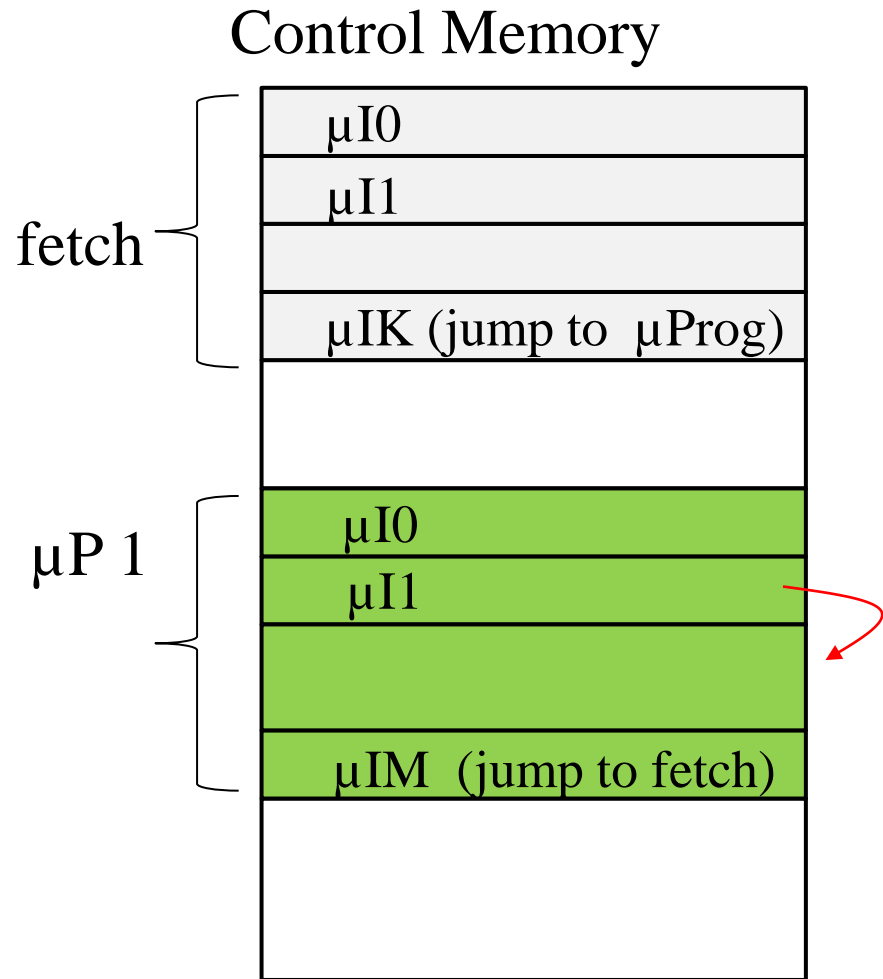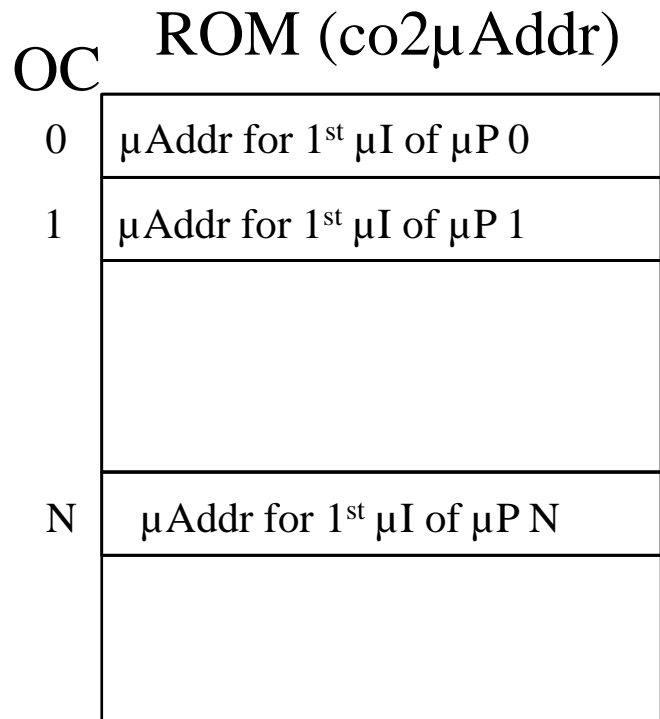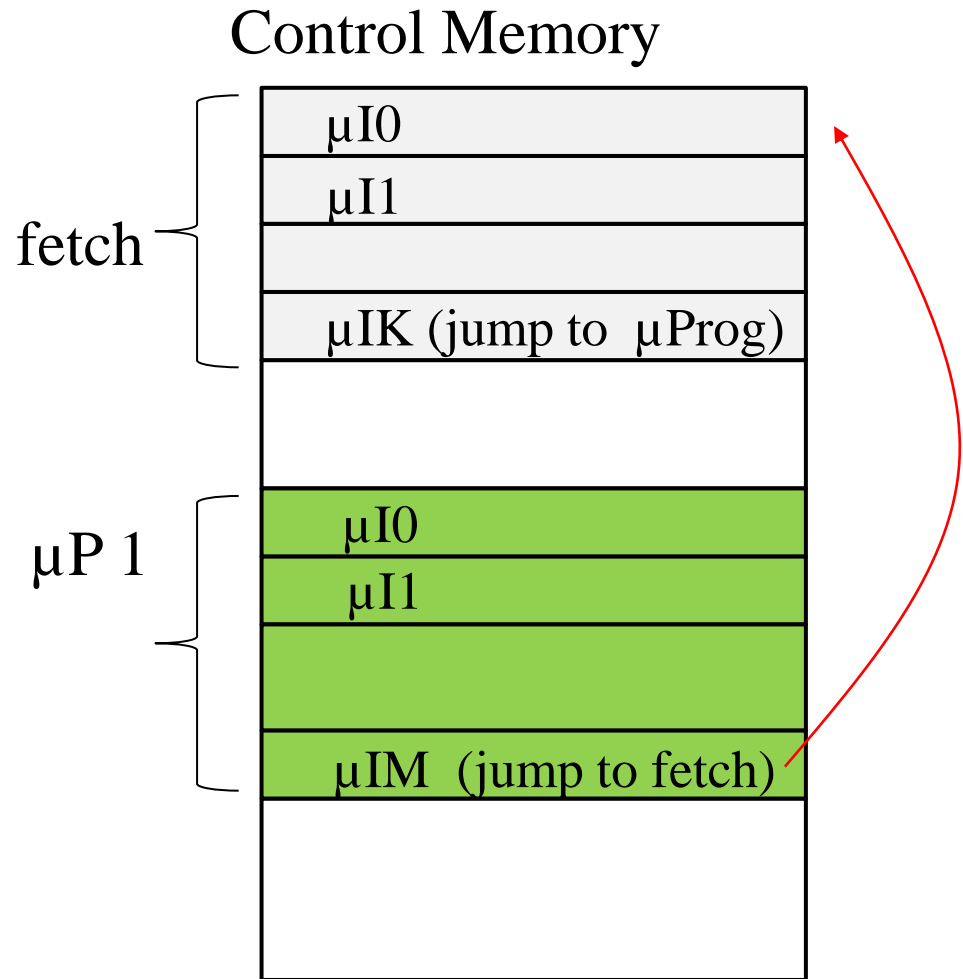
Microinstruction format

▸ **Microinstruction format**:
specifies the number of bits and the meaning of each bit.
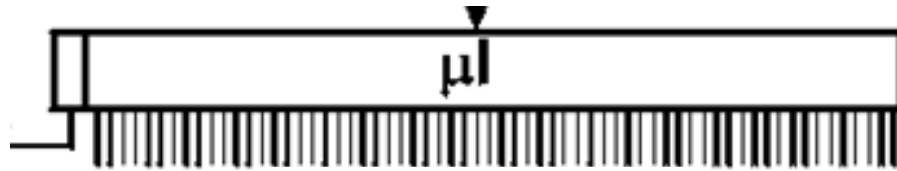
Control signals

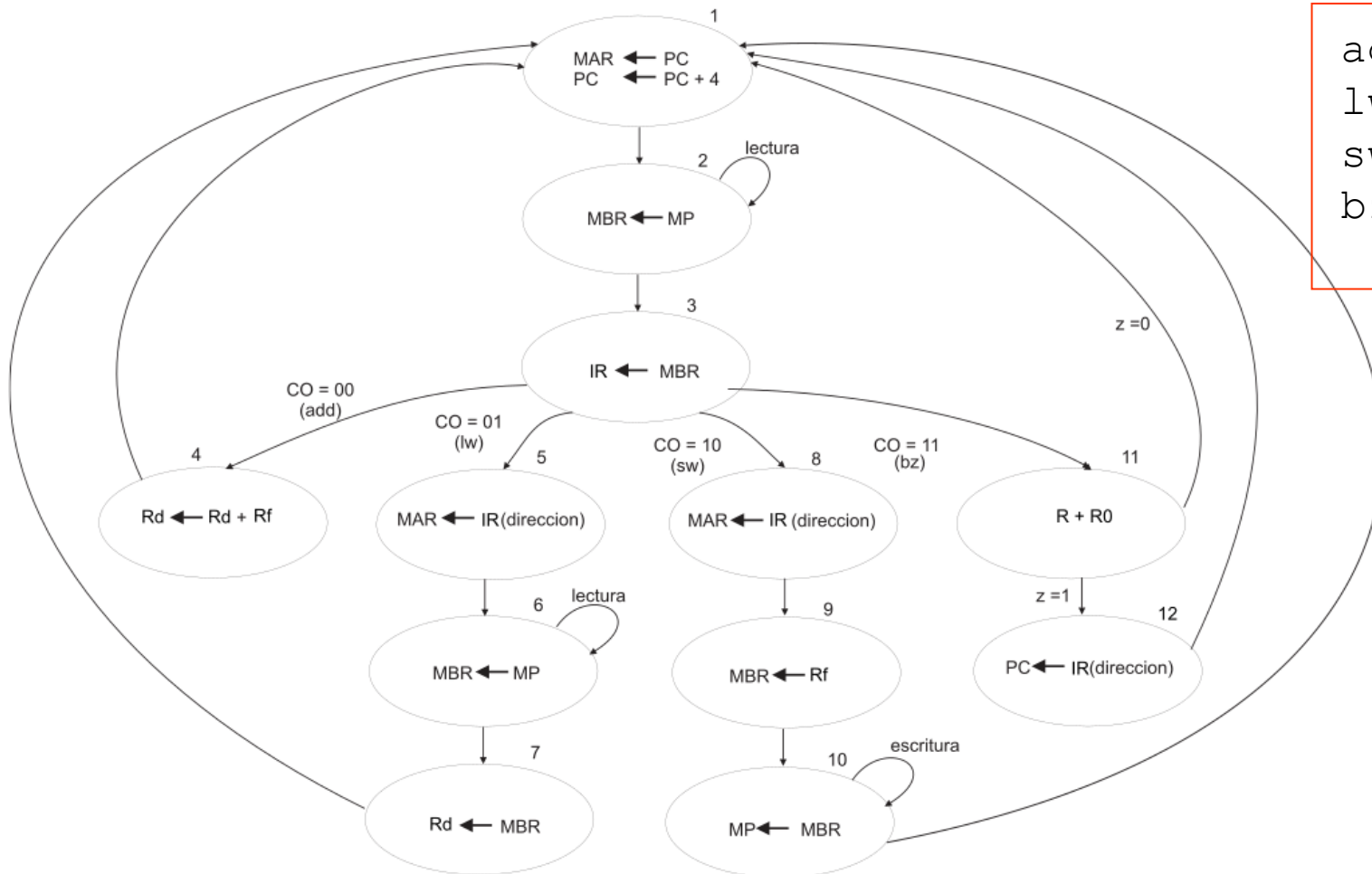▸ Signals grouped into **fields**:
  ▸ Tristate bus signals
  ▸ ALU signals
  ▸ Registers file signals
  ▸ Main memory signals
  ▸ Multiplexor signals

Example of state machine

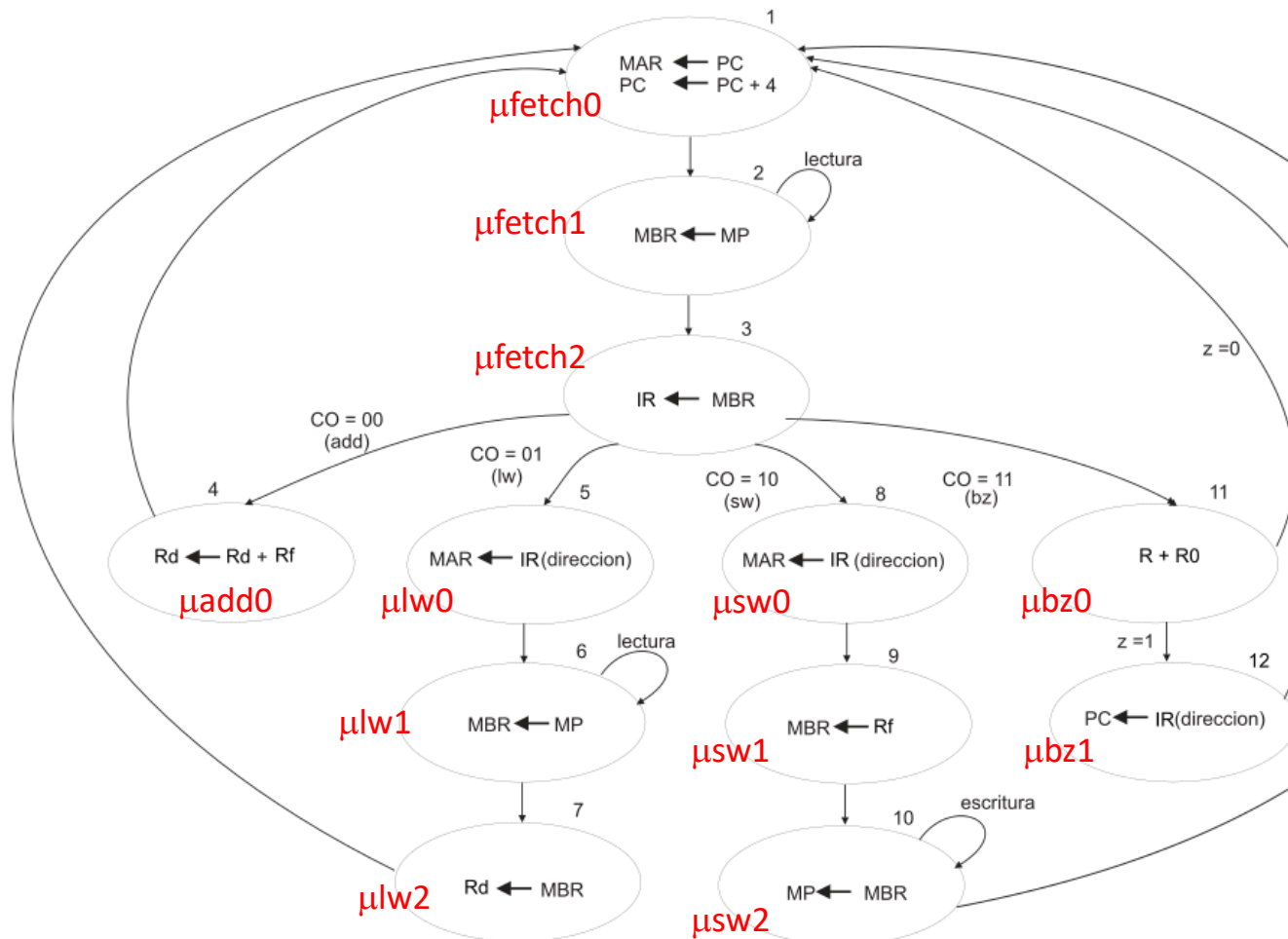Example for a computer with only 4 machine instructions

```
add rd, rf
lw rd, dir
sw Rf, dir
bz R, dir
```

Microinstructions for the example

Example for a computer with only 4 machine instructions
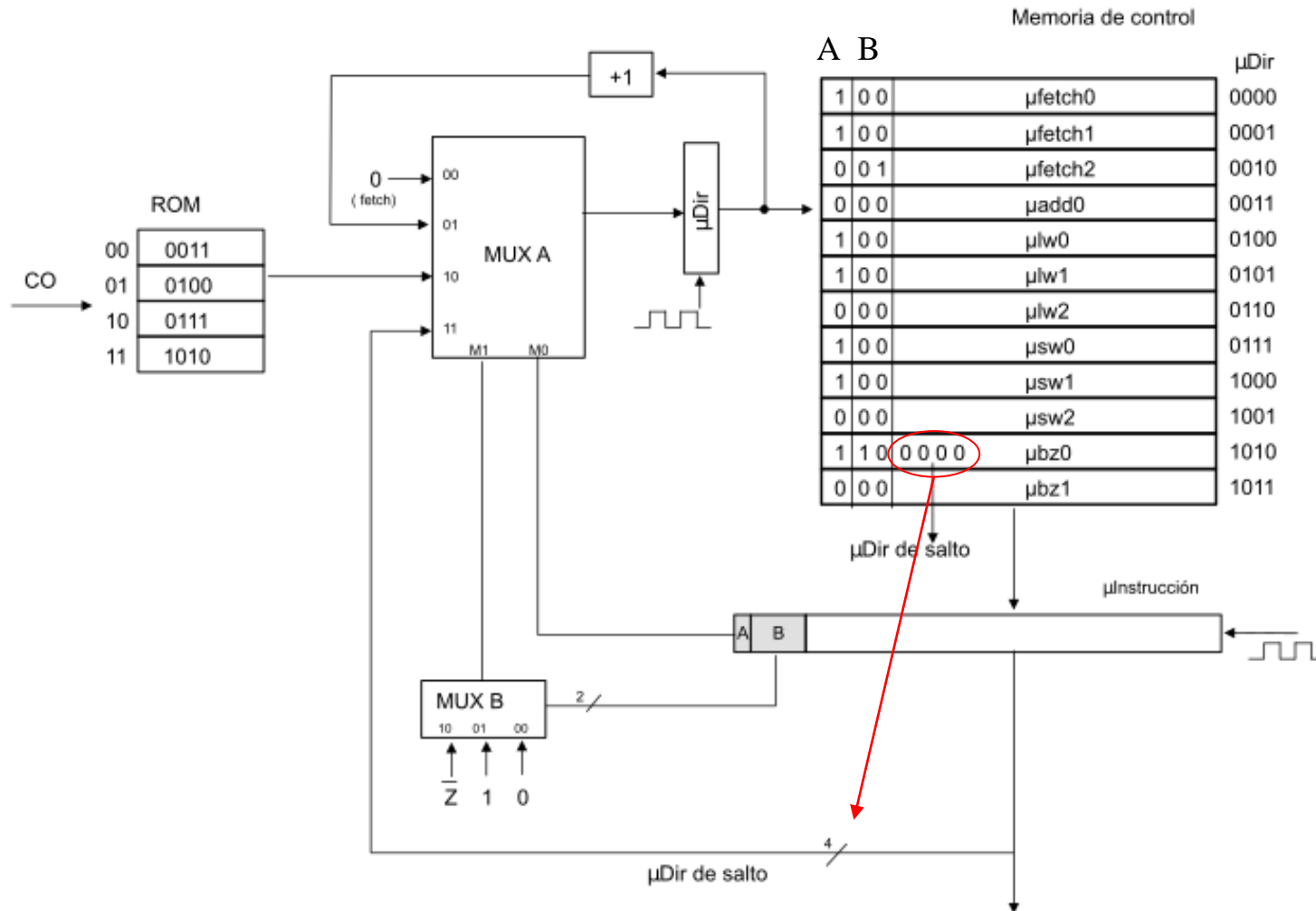
```
add rd, rf
lw rd, dir
sw Rf, dir
bz R, dir
```

ARCOS @ UC3M
Félix García-Carballeira, Alejandro Calderón Mateos

Microcode for the example

```
add r1, r2
lw r1, dir
bz dir
sw r1
```

	C0	C1	C2	C3	C4	C5	C6	C7	T1	T2	T3	T4	T5	T6	T7	T8	T9	T10	LE	MA	MB1	MB0	M1	M2	M7	R	W	Ta	Td	
µfetch0	1	0	1	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	fetch
µfetch1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1	0	1	0	
µfetch2	0	0	0	1	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
µadd0	0	0	0	0	0	0	0	1	0	0	0	0	0	1	0	0	0	0	1	0	0	0	0	1	0	0	0	0	0	add
µlw0	1	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	lw
µw1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1	0	1	0	
µlw2	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	
µsw0	1	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	sw
µsw1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	1	0	0	0	0	0	0	
µsw3	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	
µbz0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	bz
µbz1	0	0	1	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	

Microprogrammed control unit for the example

ARCOS @ UC3M
Félix García-Carballeira, Alejandro Calderón Mateos
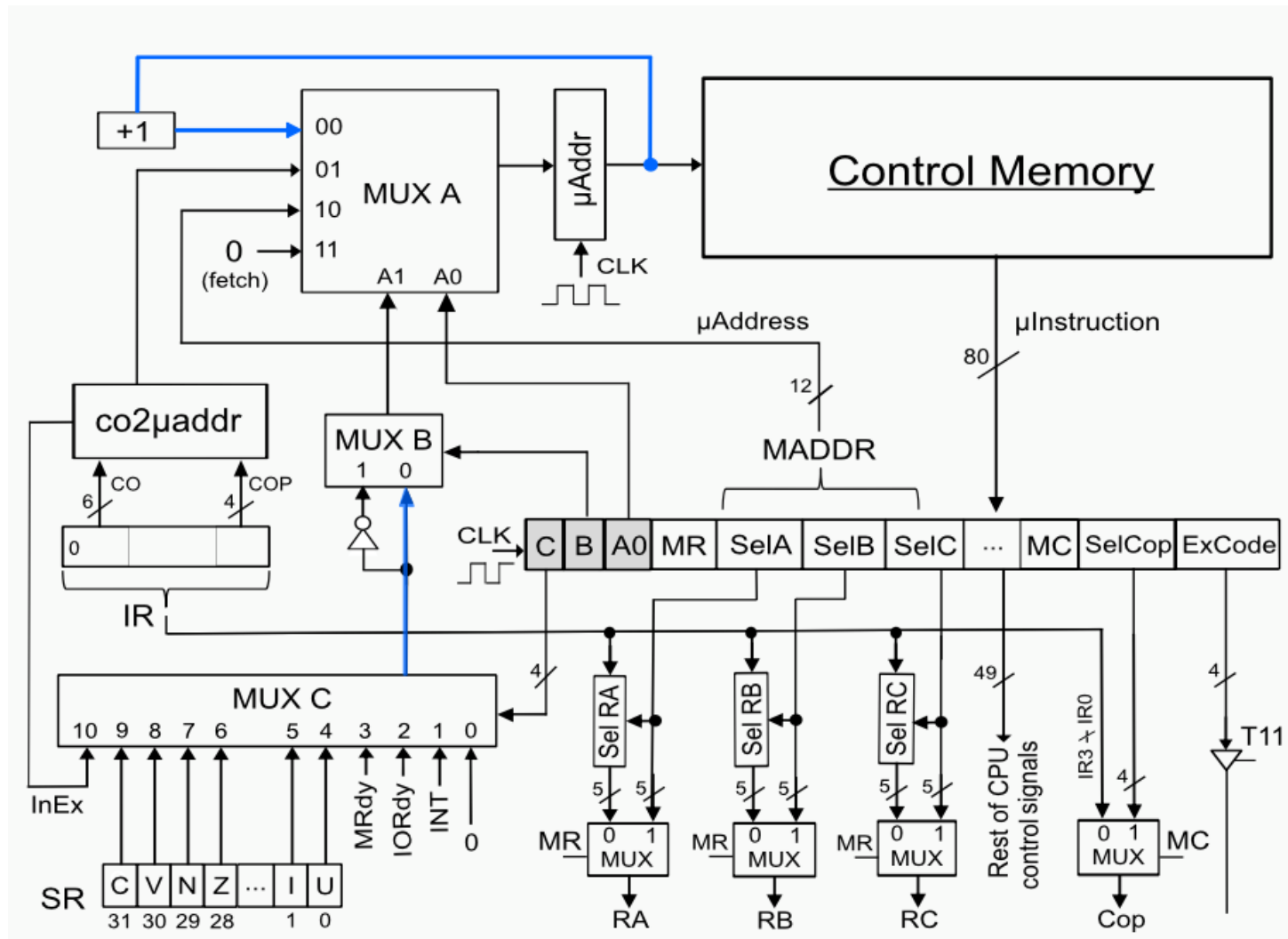
Non-existing instruction
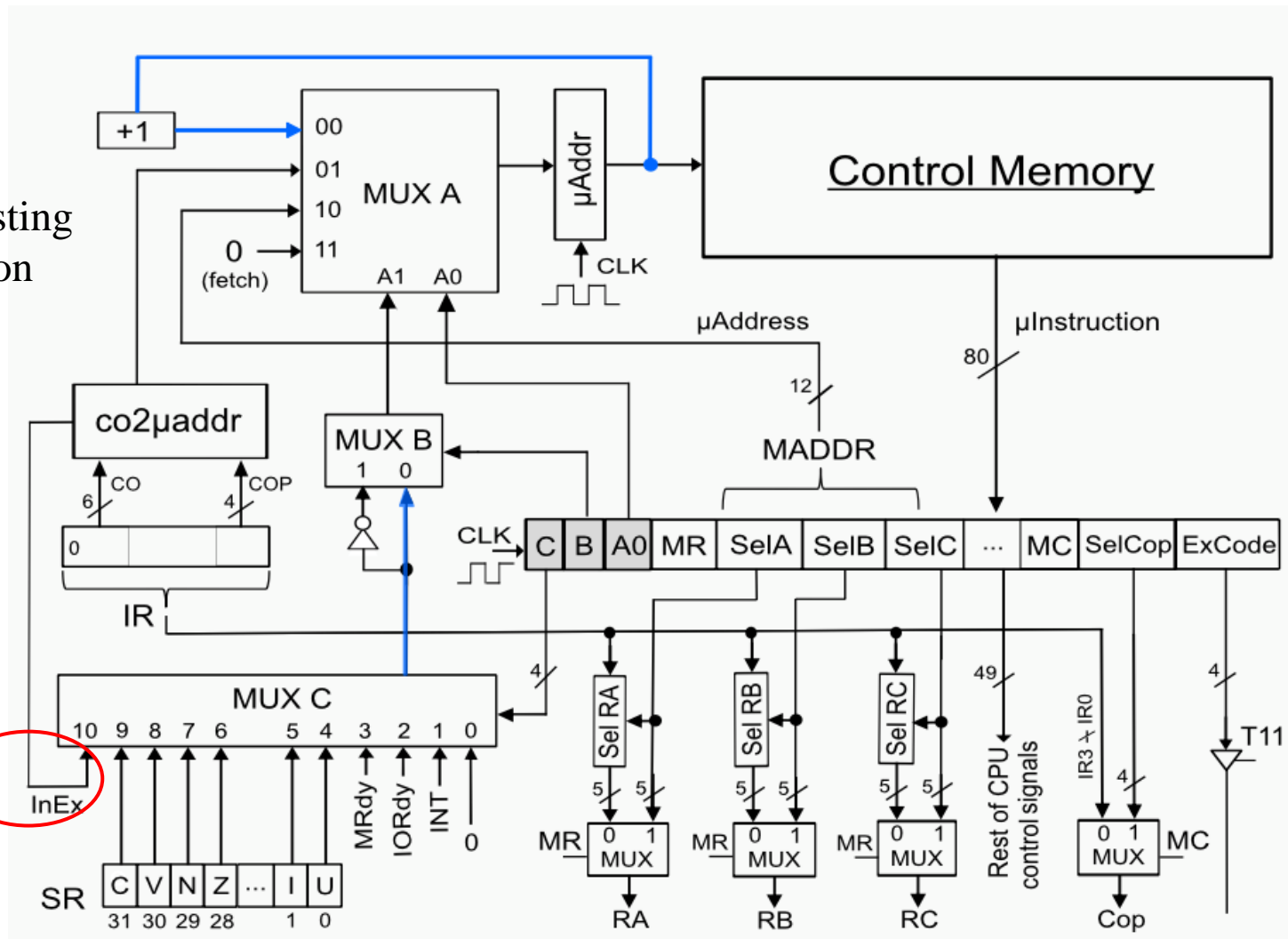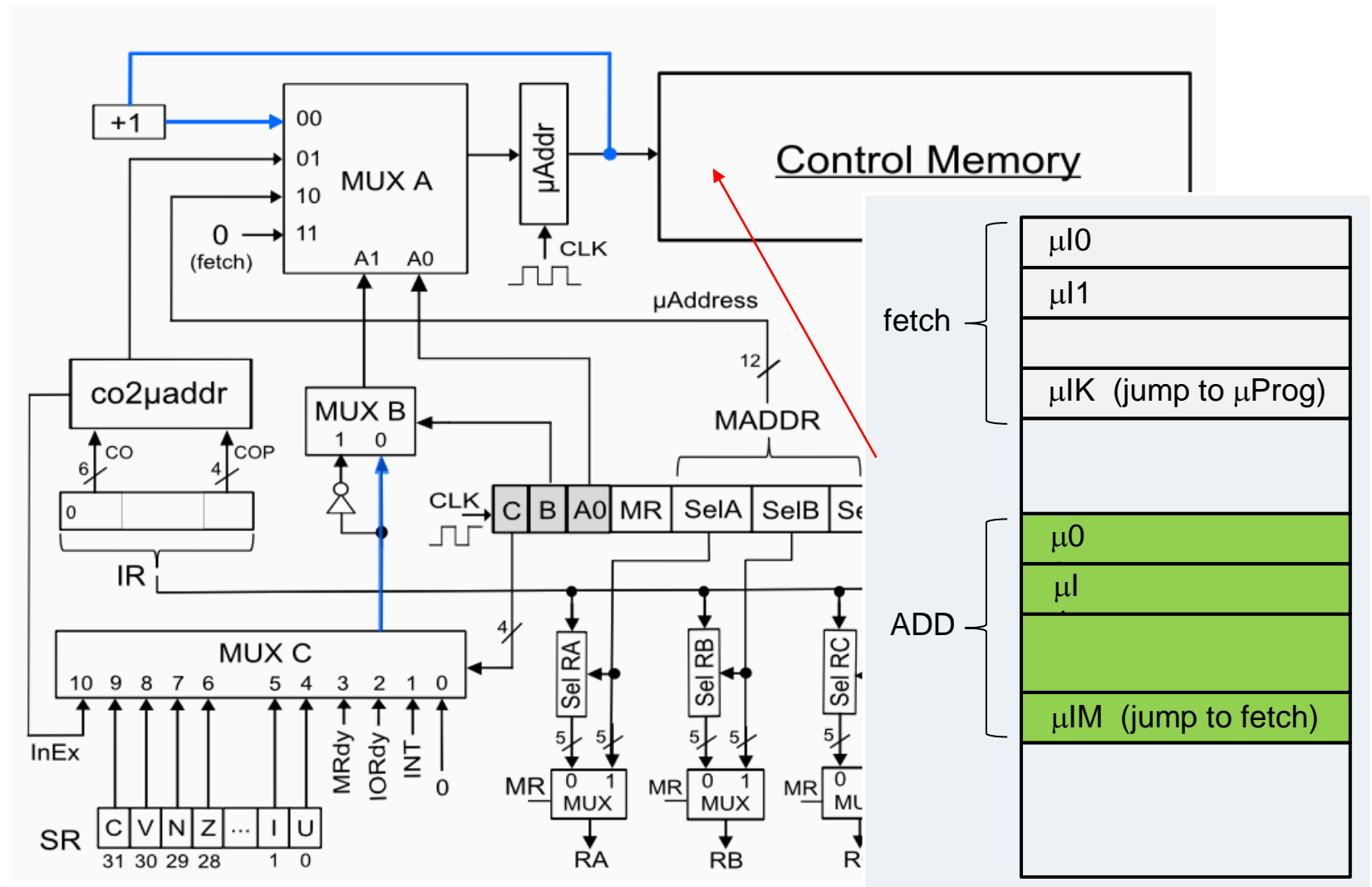
Control Unit in WepSIM
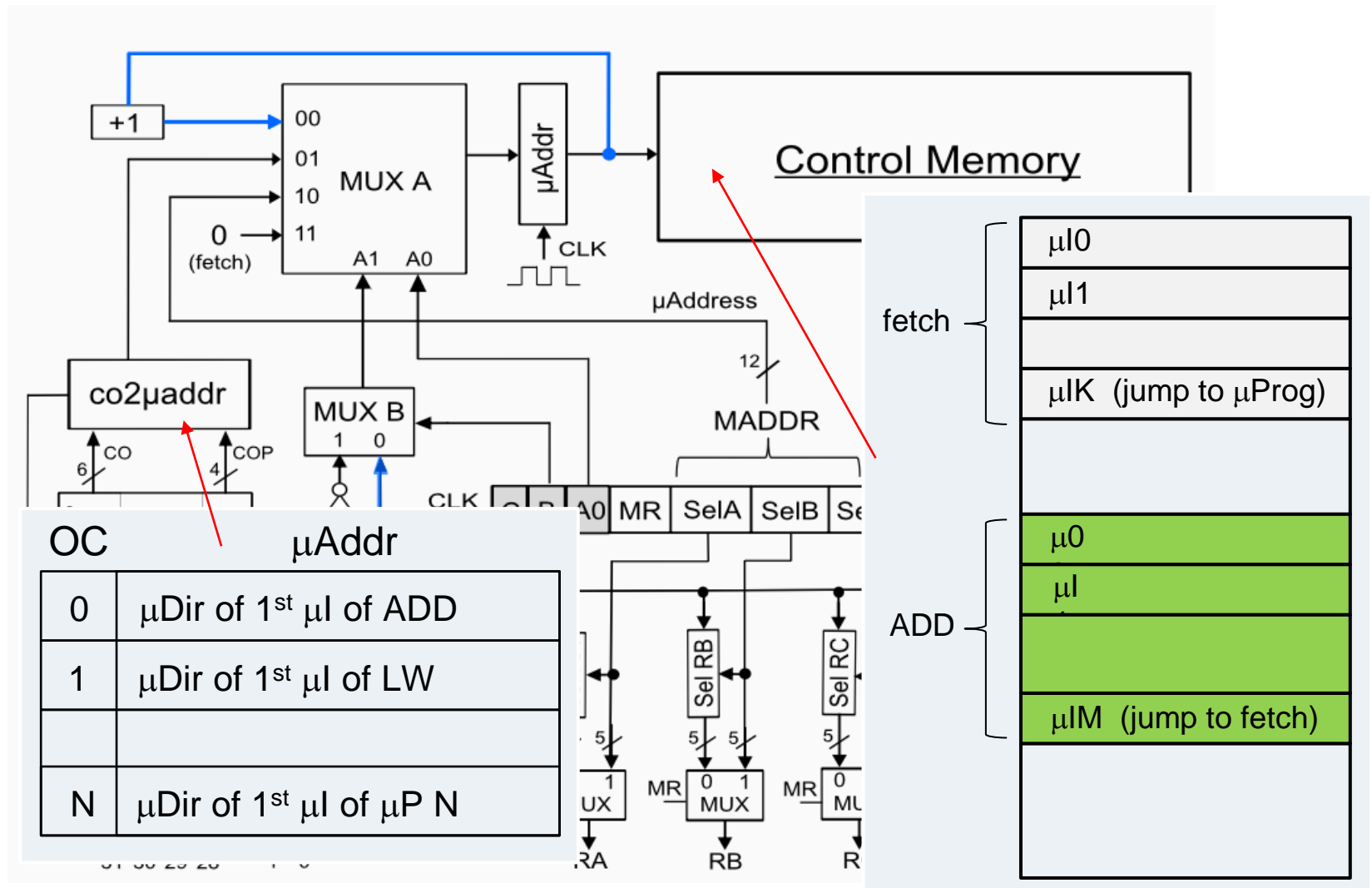
Control Unit in WepSIM

A1	A0	Action
0	0	Next μDir
0	1	Jump to μProg. (ROM)
1	0	μDir of jump
1	1	fetch
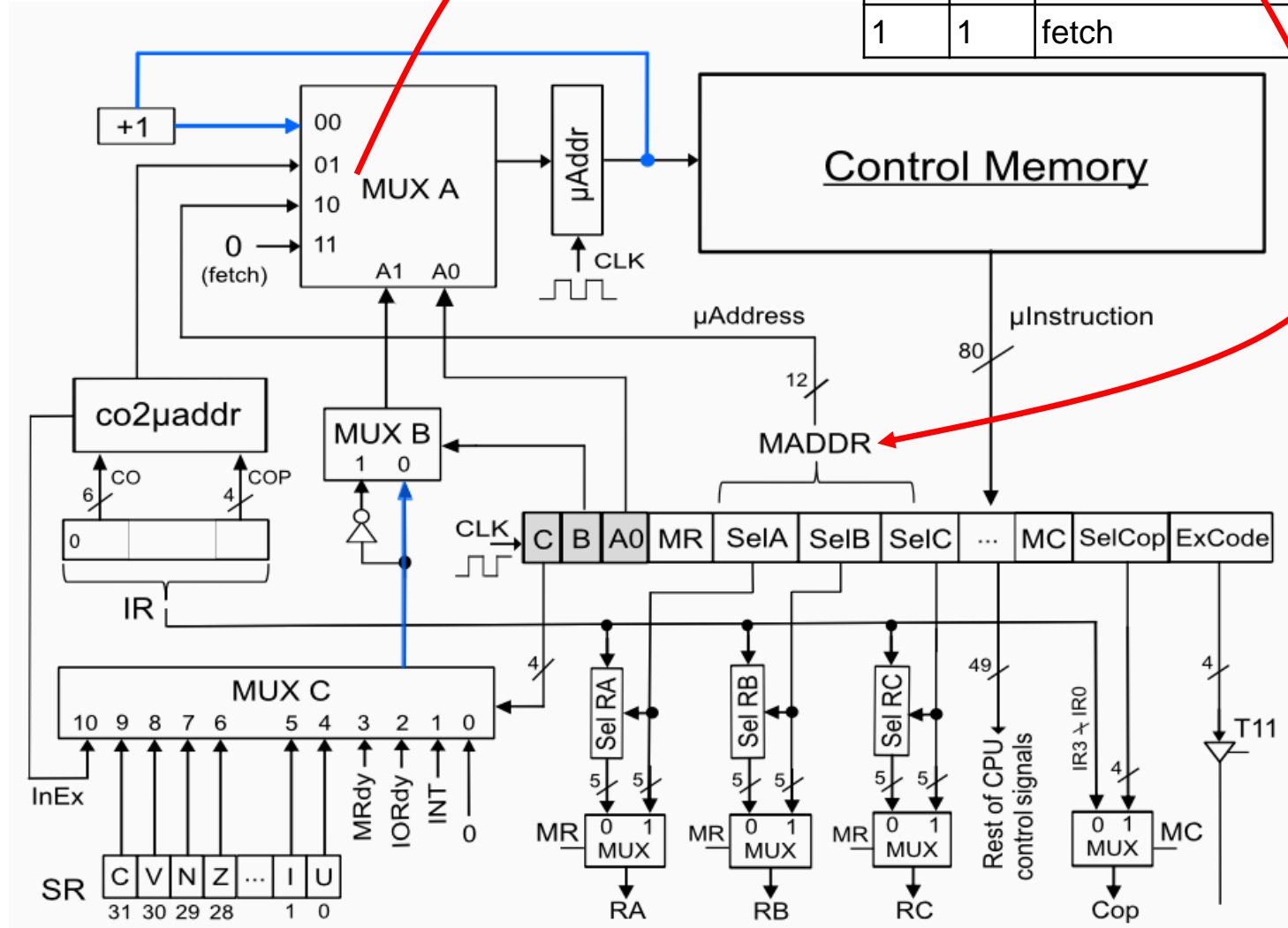
Control Unit in WepSIM

A0	B	C3	C2	CI	C0	Action
0	0	0	0	0	0	Next µAddress
0	I	0	0	0	0	Unconditional jump to MADDR
0	0	0	0	0	I	Conditional jump to MADDR if INT = I (*)
0	I	0	0	I	0	Conditional jump to MADDR if IORdy = 0 (*)
0	I	0	0	I	I	Conditional jump to MADDR if MRdy = 0 (*)
0	0	0	I	0	0	Conditional jump to MADDR if U = I (*)
0	0	0	I	0	I	Conditional jump to MADDR if I = I (*)
0	0	0	I	I	0	Conditional jump to MADDR if Z = I (*)
0	0	0	I	I	I	Conditional jump to MADDR if N = I (*)
0	0	I	0	0	0	Conditional jump to MADDR if O = I (*)
I	0	0	0	0	0	Jump to µProg. (ROM c02µaddr)
I	I	0	0	0	0	Jump to fetch (µDir = 0)

▸ (*) If the condition is not satisfied → Next µAddress
▸ Rest of entries → undefined behavior

Example
Elemental operations with CU

▸ **Jump to address 000100011100 (12 bits) if Z = 1. Otherwise jump to the next one.**

Elemental operation	Signals
If (Z) µPC=000100011100	A0=0, B=0, C=0110_2, mADDR=000100011100_2

▸ **Unconditional jump to address 000100011111**

Elemental operation	Signals
µPC=000100011111	A0=0, B=1, C=0000_2, mADDR=000100011111_2

▸ **Jump to first µaddress of the µprogram related to OC**

Elemental operation	Signals
Jump to OC	A0=1, B=0, C=0000_2

Example

▸ Jump to the µAddress 000100011100 (12 bits) if Z = 1. Otherwise, jump to the next one:

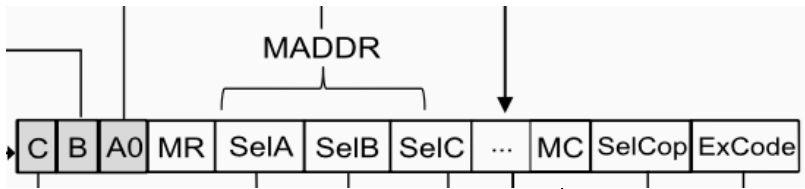- ▸ A0 = 0
- ▸ B = 0
- ▸ C = 0110
- ▸ µAddr = 000100011100
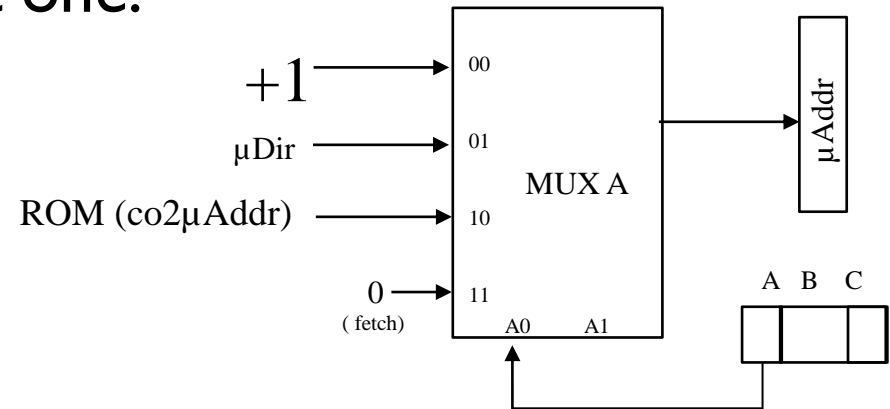
▸ Unconditional jump to µAddress 000100011111

- ▸ A0 = 0
- ▸ B = 1
- ▸ C = 0000
- ▸ µAddr = 000100011111

µAddress encoded
in bits 72-61
of the µInstruction

Microinstruction format

MADDR

| C | B | A0 | MR | SelA | SelB | SelC | ... | MC | SelCop | ExCode |

~80 bits

C0 .. C7	Load register
Ta, Td	Tristate buffers to bus
T1..T10	Tristate buffers
M1,M2, M7, MA, MB	Multiplexors
SelP	State register selector
LC	Load in Register File
SE	Sign extensión
Size, Offset	Selector of IR register
BW	Size of memory Access
R, W	Main memory operation
IOR, IOW	I/O operation
INTA	INT selector
I	Enables interuptions
U	User/kernel modes

Register file selector
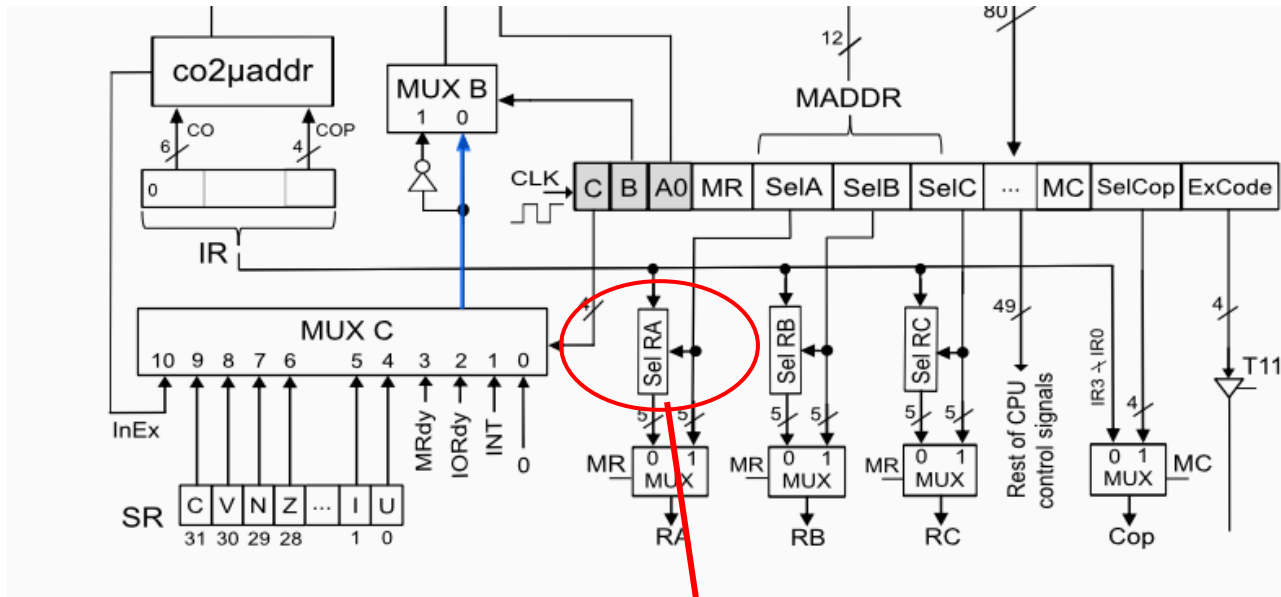
Select 5 bits within 32-bit starting from the position indicated in *Displ* (lower bit)

Data (IR)

32

Selector

5 — Displ (SelA, SelB, SelC)

5
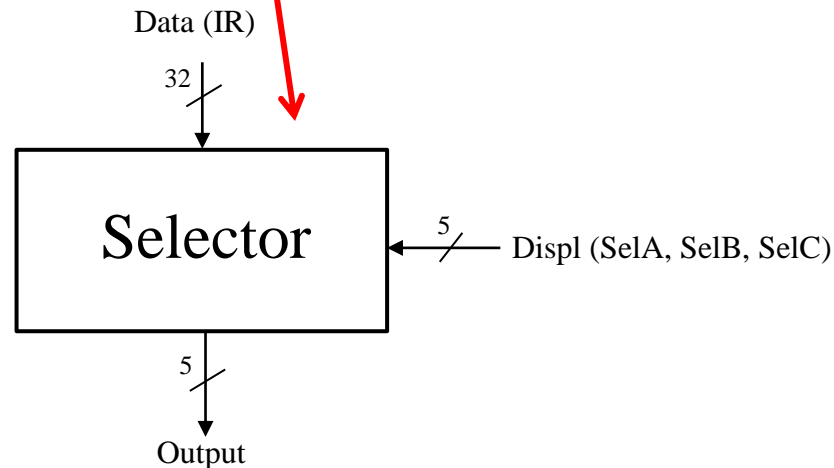
Output

Register file selector
Example

Data (IR)

32

```
Selector
```

5 — Displ

5
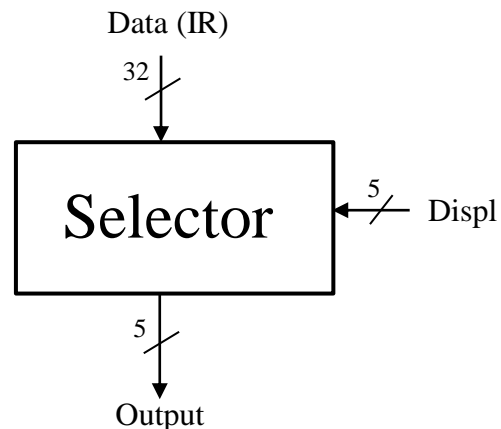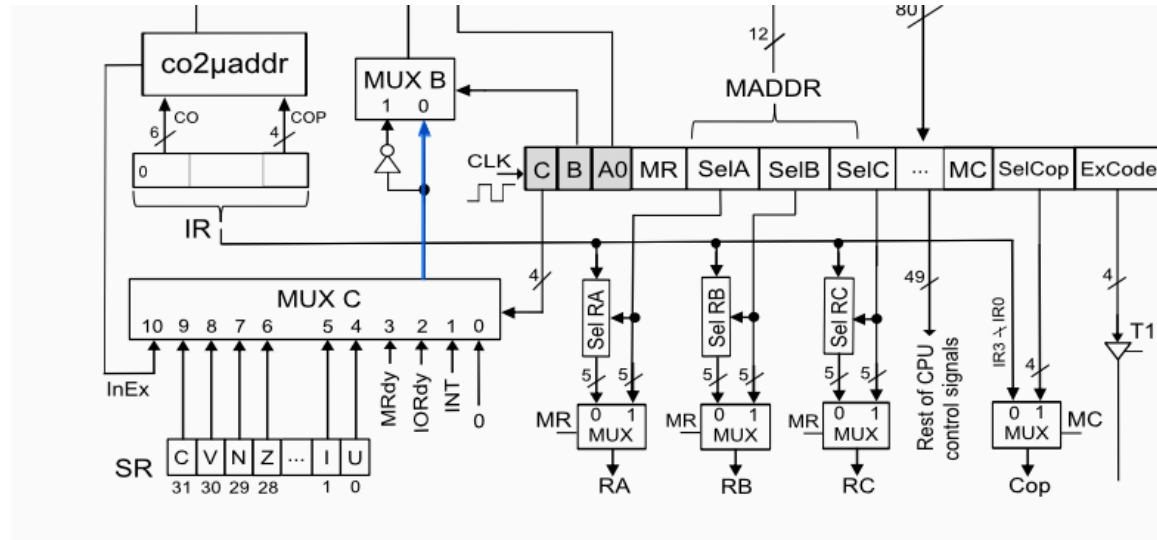
Output

RI: $D_{31}D_{30}D_{29}D_{28}D_{27}D_{26}D_{25}$ ……. $D_4D_3D_2D_1D_0$

If Displ = 11011   →   Output = $D_{31}D_{30}D_{29}D_{28}D_{27}$

If Displ = 00000   →   Output = $D_4D_3D_2D_1D_0$

If Displ = 10011   →   Output = $D_{23}D_{22}D_{21}D_{20}D_{19}$

If Displ = 01011   →   Output = $D_{15}D_{14}D_{13}D_{12}D_{11}$
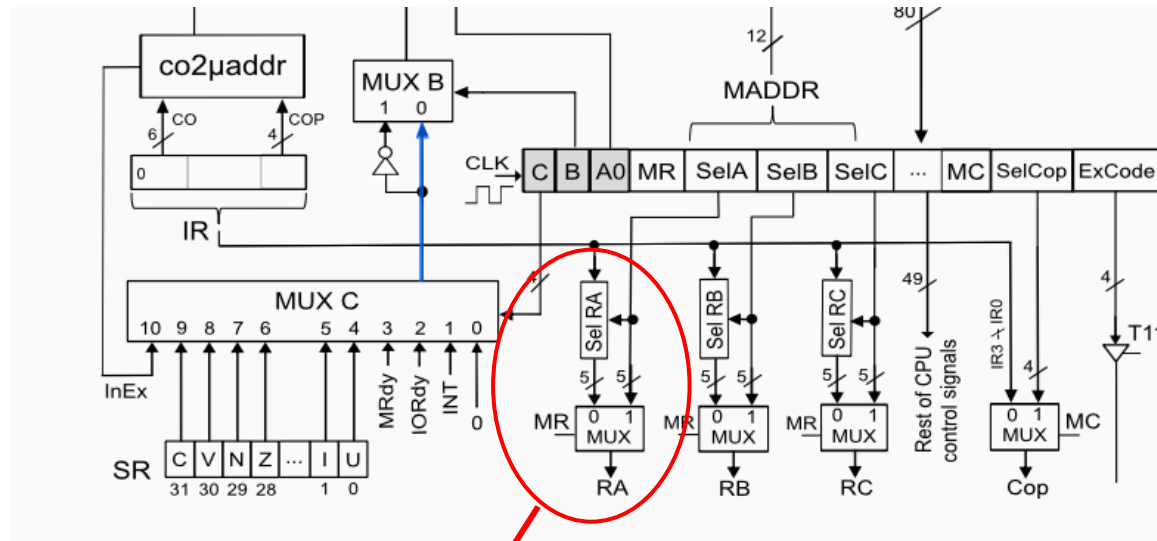
ARCOS @ UC3M
Félix García-Carballeira, Alejandro Calderón Mateos

Register file selector

Select 5 bits within 32-bit starting from the position indicated in *Displ* (lower bit)

Data (IR)

32

Selector ← 5 — Displ

5

Output
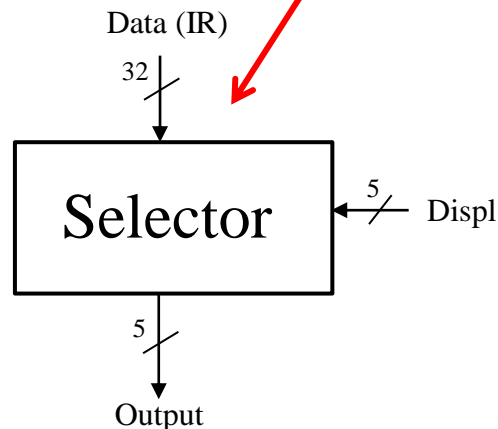
- If MR = 1,   RA is obtained from the µInstruction

- If MR = 0,   RA is obtained from a field of IR

ARCOS @ UC3M
Félix García-Carballeira, Alejandro Calderón Mateos

Register file selector

▸ If the format of an instruction stored in IR is:

31	26	25	21	20	16	15	11	10	0	
CO_5 ... CO_0										Instruction Register (IR)
Operation Code		Reg 1		Reg 2		Reg 3				
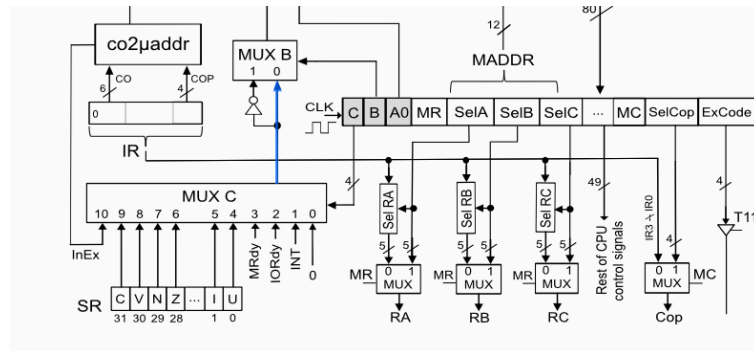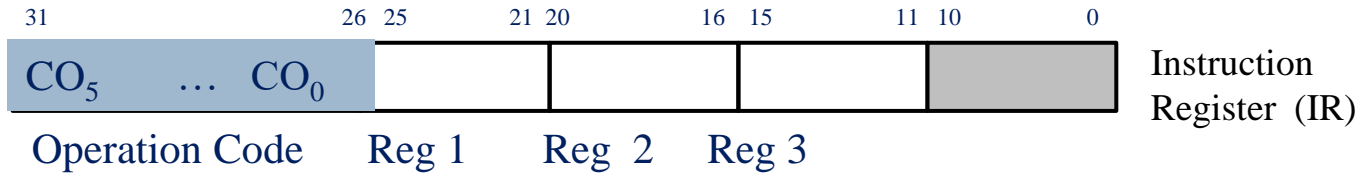
$$MR = 0$$

▸ If you want to select the field with the Reg 2 in gate B of the record file → SelB = 10000 (RB is obtained from bits 20...16 of IR)

▸ If you want to select the field with the Reg 3 in port A of the record file → SelA = 01011 (RA is obtained from bits 15...11 of IR)

▸ If you want to select the field with the Reg 1 in gate C of the record file → SelC = 10101 (RC is obtained from bits 25...21 of IR)

Selection of the ALU operation
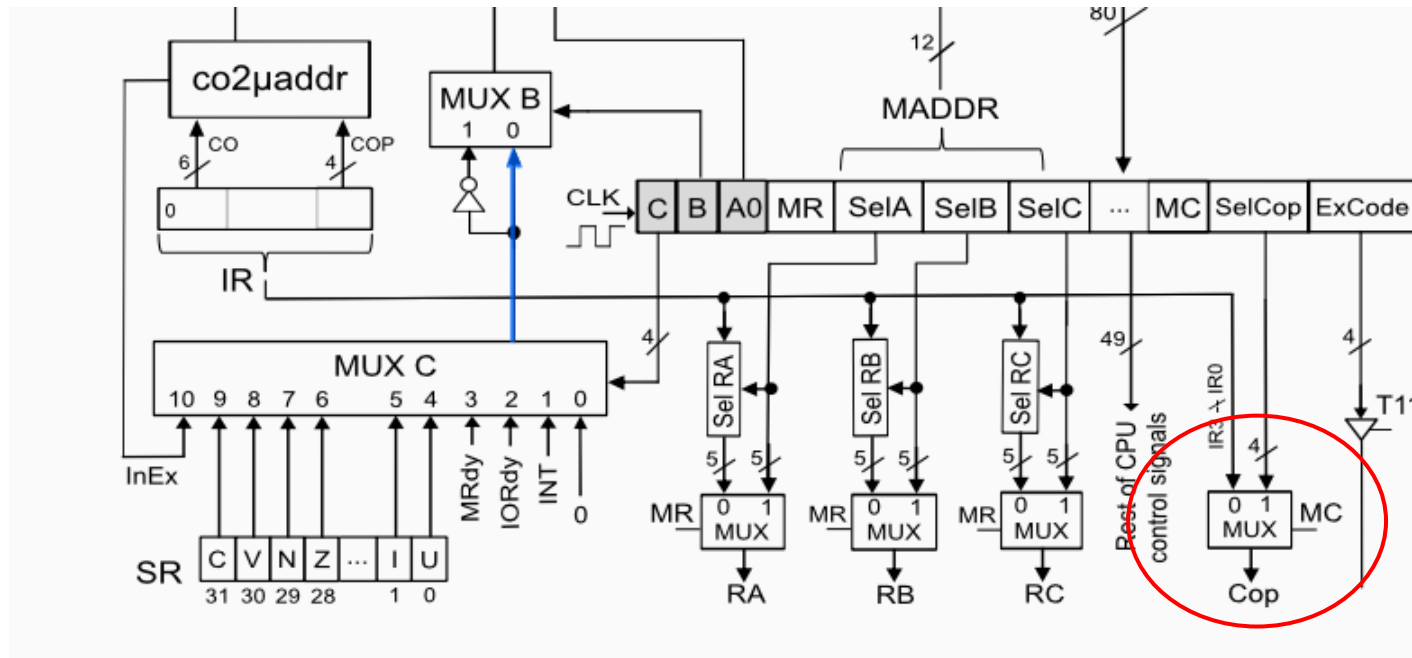
- If MC = 1, the operation code of the ALU is obtained directly from the microinstruction (SelCop)
- If MC = 0, the operation code of the ALU is obtained from the last four bits stored in the instruction register

Exception codes
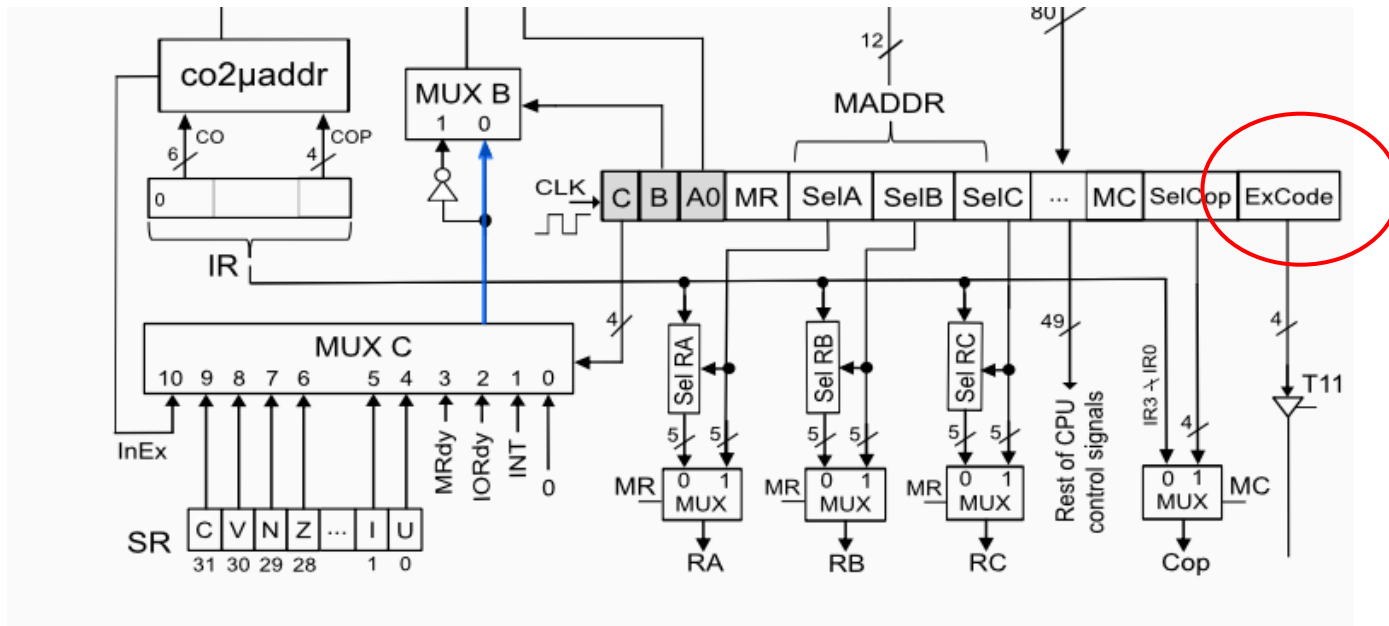
- **ExCode:**
  - Allows to have an immediate value of any 4 bits,
  - Especially useful for generating the interrupt vector to be used when an exception occurs in the instruction.

Examples

- Instruction to microprogramming with WepSIM*:

Instruction	Operation code	Meaning
ADD Rd, Rf1, Rf2	000000	Rd ← Rf1 + Rf2
LI R, value	000001	R ← value
LW R, addr	000010	R ← MP[addr]
SW R, addr	000011	MP[addr] ← R
BEQ Rf1, Rf2, off1	000100	if (Rf1 == Rf2) PC ← PC +off1
J addr	000101	PC ← addr
HALT	000110	HALT (infinite loop)

* Memoria de un ciclo

Microprogrammed instructions

▸ FETCH

Cycle	Elemental Op.	Activated signals (rest to 0)	C	B	A0
0	MAR ← PC	T2, C0	0000	0	0
1	MBR ← MP	Ta, R, BW = 11, C1, M1	0000	0	0
	PC ← PC + 4	M2, C2	0000	0	0
2	IR ← MBR	T1, C3	0000	0	0
3	Decode		0000	0	1

Microprogrammed instructions

▶ ADD Rd, Rf1, Rf2

Cycle	Elemental Op.	Activated signals (rest to 0)	C	B	A0
0	Rd ← Rf1 + Rf2	SelCop = 1010, MC SelP=11, C7, M7 T6, LC SelA = 10000 (16) SelB = 01011 (11) SelC = 10101 (21)	0000	1	1

6 bits	5 bits	5 bits	5 bits	11 bits
O.C.	Rd	Rf	Rf2	not used

31　　　　　　　26 25　　　　　21 20　　　　　16 15　　　　　11 10　　　　　　　　　0
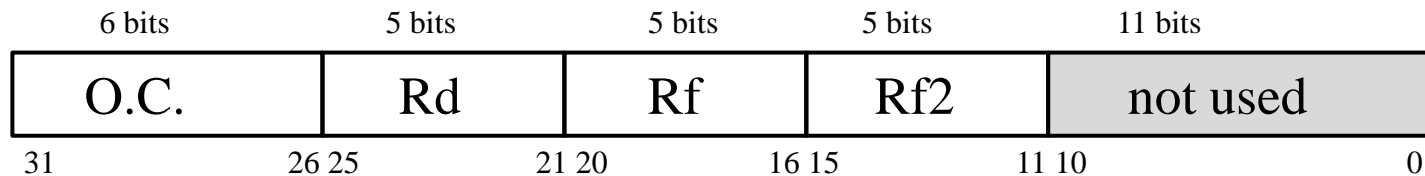
Microprogrammed instructions (another)

▸ ## ADD Rd, Rf1, Rf2

Cycle	Elemental Op.	Activated signals (rest to 0)	C	B	A0
0	Rd ← Rf1 + Rf2	SelCop = 1010, MC SelP=11, C7, M7 T6, LC SelA = 10000 (16) SelB = 01011 (11) SelC = 10101 (21)	0000	1	1

6 bits	5 bits	5 bits	5 bits	7 bits	4 bits
000000	Rd	Rf1	Rf2	not used	1001

31   26 25   21 20   16 15   11 10   4 3   0
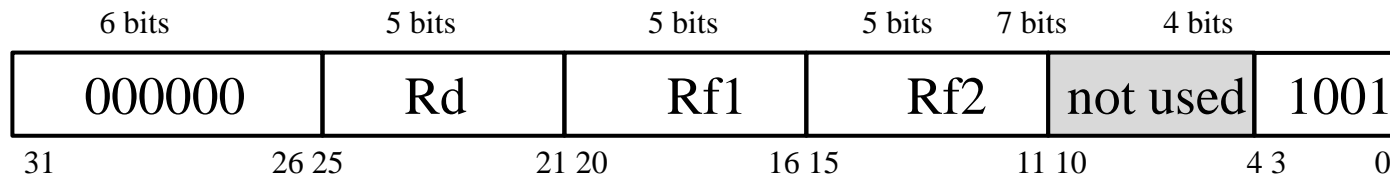
Microprogrammed instructions

▶ LI R, value

Cycle	Elemental Op.	Activated signals (rest to 0)	C	B	A0
0	R ← IR (value)	LC SelC = 10101 (21) T3, Size = 10000 Offset= 00000 SE=1	0000	1	1

	6 bits	5 bits	5 bits	16 bits	
	O.C.	R	not used	value of 16 bits	

31   26 25   21 20   16 15   0
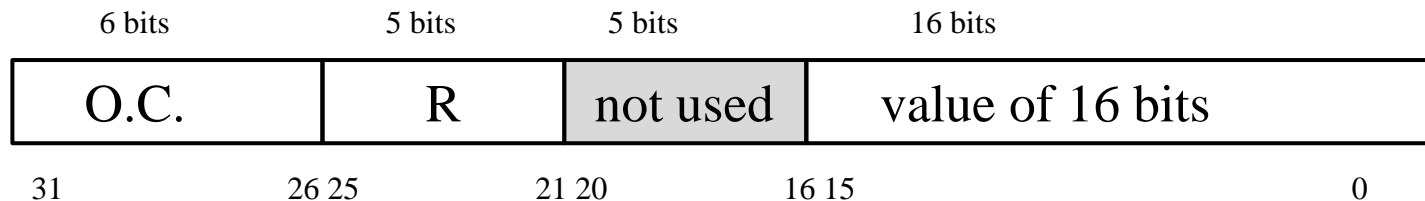
Microprogrammed instructions

▸ LW R addr   # sync memory, 1 clock cycle

Cycle	Elemental Op.	Activated signals (rest to 0)	C	B A0
0	MAR ← IR (dir)	T3, C0 Size = 10000, Offset= 00000	0000	0 0
1	MBR ← MP[MAR]	Ta, R, BW =11, C1, M1	0000	0 0
2	R ← MBR	T1, LC, SelC = 10101	0000	1 1

6 bits	5 bits	5 bits	16 bits
O.C.	R	not used	address of 16 bits

31   26 25   21 20   16 15   0
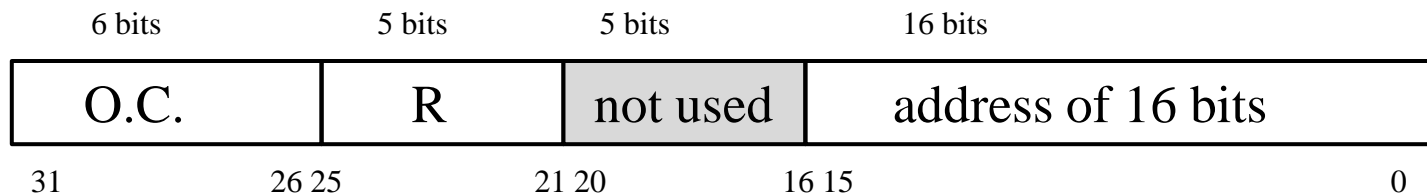
Microprogrammed instructions

▸ LW R addr   # async memory (MRdy=1 for ready)

Cycle	Elemental Op.	Activated signals (rest to 0)	C	B	A0
0	MAR ← IR (dir)	T3, C0 Size = 10000, Offset= 00000	0000	0	0
1	while (!MRdy) MBR ← MP[MAR]	Ta, R, BW =11, C1, M1, MADDR=µAdd of this microinstruction	0011	1	0
2	R ← MBR	T1, LC, SelC = 10101	0000	1	1

This microinstruction is beening executed while MRdy==0

Microprogrammed instructions

▶ SW R addr   # sync memory, 1 clock cycle

Cycle	Elemental Op.	Activated signals (rest to 0)	C	B	A0
0	MBR ← R	T9, C1, SelA=10101	0000	0	0
1	MAR ← IR(addr)	T3, C0, Size = 10000, offset= 00000	0000	0	0
2	MP[addr] ← MBR	Td, Ta, BW = 11, W	0000	1	1

6 bits	5 bits	5 bits	16 bits
O.C.	R	not used	address of 16 bits

31   26 25   21 20   16 15   0
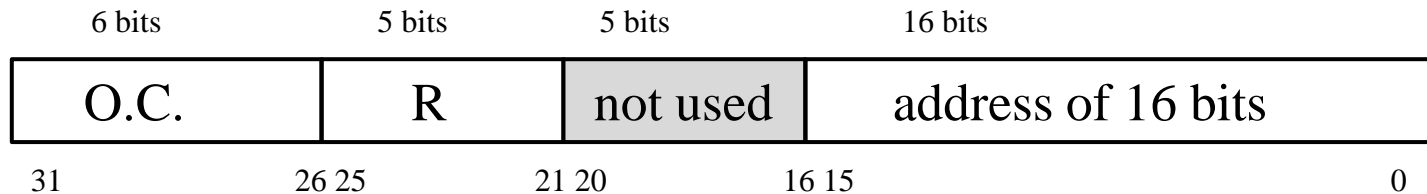
Microprogrammed instructions

▸ BEQ Rf1, Rf2, offset1

Cycle	Elemental Op.	Activated signals (rest to 0)	C	B	A0
0	Rf1- Rf2	SelCop = 1011, MC, C7, M7 SelP = 11, SelA = 10101 SelB = 10000	0000	0	0
11	If (Z == 0) goto fetch else next	MADDR = 0	0110	1	0
2	RT1 ←PC	T2, C4	0000	0	0
3	RT2 ← IR (offset1)	Size = 10000 Offset = 00000, T3,C5	0000	0	0
4	PC ← RT1 +RT2	SelCop = 1010, MC, MA, MB=01, T6,C2,	0000	1	1

	6 bits	5 bits	5 bits	16 bits
	O.C.	Rf1	Rf2	offset1
	31 26	25 21	20 16	15 0
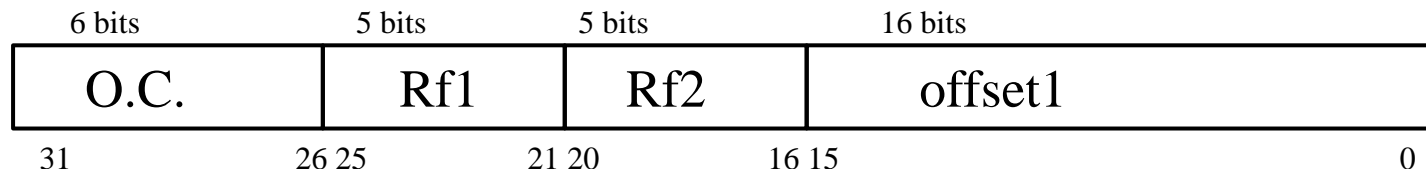
Microprogrammed instructions

▸ J dir

Cycle	Elemental Op.	Activated signals (rest to 0)	C B A0
0	PC ← IR (dir)	C2, T3, size = 10000, offset= 00000	0000 1 1

6 bits	10 bits	16 bits
O.C.	not used	address of 16 bits

31   26 25   16 15   0

Defining instructions in WepSIM

<List of implemented instructions>

<Register file specification>

<Pseudo instructions>

ARCOS @ UC3M
Félix García-Carballeira, Alejandro Calderón Mateos

Defining instructions in WepSIM

```
begin
{
        fetch:      (T2, C0=1),
                    (Ta, R, BW=11, C1, M1),
                    (M2, C2, T1, C3),
                    (A0, B=0, C=0)
}
```

ARCOS @ UC3M
Félix García-Carballeira, Alejandro Calderón Mateos

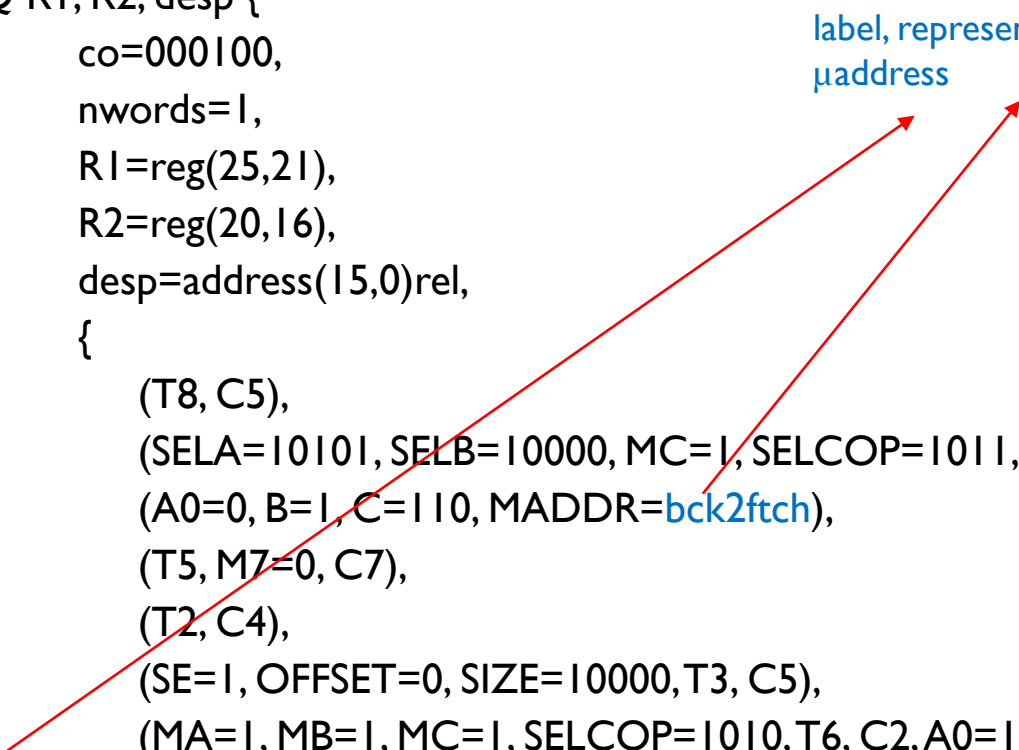# Defining instructions in WepSIM

```
ADD R1,R2, R3 {
        co=000000,
        nwords=1,
        R1=reg(25,21),
        R2=reg(20,16),
        R3=reg(15,11),
        {
                (SelCop=1010, MC, SelP=11,  M7,C7,T6, LC,
                 SelA=01011, SelB=10000, SelC=10101,
                 A0=1, B=1, C=0)
        }
}
```

ARCOS @ UC3M
Félix García-Carballeira, Alejandro Calderón Mateos

Defining instructions in WepSIM

```
BEQ R1, R2, desp {
        co=000100,
        nwords=1,
        R1=reg(25,21),
        R2=reg(20,16),
        desp=address(15,0)rel,
        {
            (T8, C5),
            (SELA=10101, SELB=10000, MC=1, SELCOP=1011, SELP=11, M7, C7),
            (A0=0, B=1, C=110, MADDR=bck2ftch),
            (T5, M7=0, C7),
            (T2, C4),
            (SE=1, OFFSET=0, SIZE=10000, T3, C5),
            (MA=1, MB=1, MC=1, SELCOP=1010, T6, C2, A0=1, B=1, C=0),
    bck2ftch: (T5, M7=0, C7),
            (A0=1, B=1, C=0)
        }
}
```

label, represents a µaddress

Félix García-Carballeira, Alejandro Calderón Mateos

Register specification

```
registers {
         0=$zero,   1=$at,
         2=$v0,     3=$v1,
         4=$a0,     5=$a1,
         6=$a2,     7=$a3,
         8=$t0,     9=$t1,
         10=$t2,  11=$t3,
         12=$t4,  13=$t5,
         14=$t6,  15=$t7,
         16=$s0,  17=$s1,
         18=$s2,  19=$s3,
         20=$s4,  21=$s5,
         22=$s6,  23=$s7,
         24=$t8,  25=$t9,
         26=$k0,  27=$k1,
         28=$gp,  29=$sp (stack_pointer),
         30=$fp,  31=$ra
}
```