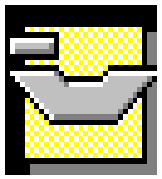


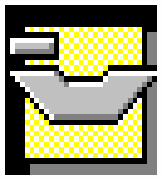
COMPUTERS ORGANIZATION

2ND YEAR COMPUTE SCIENCE MANAGEMENT ENGINEERING

UNIT 2 – CONTROL UNIT

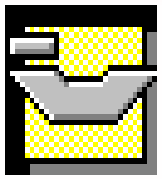
JOSÉ GARCÍA RODRÍGUEZ
JOSÉ ANTONIO SERRA PÉREZ

departamento de tecnología informática y computación
UNIVERSIDAD DE ALICANTE



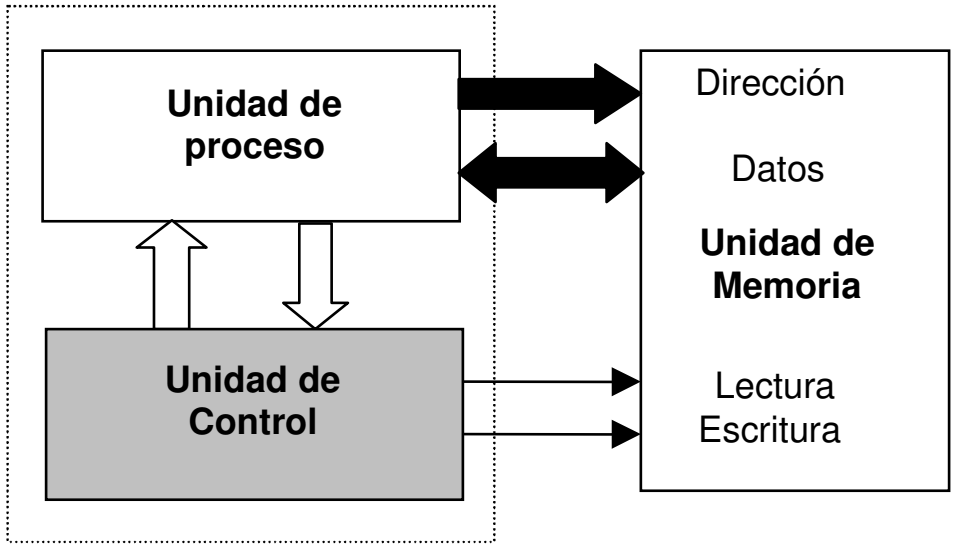
Central Process Unit

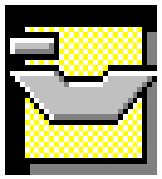
- Introduction
- Buses use
- Stages in the Instructions Execution
  - Introduction
  - Establishment of the stages
  - Data path
- Control Unit
  - Control signals identification
  - Control signals activation
- Design of the Control Unit
  - States Table Method
  - Sequence Counter Method
  - Machine abstraction
- Conclusions



Introduction

- Way of working of the instructions execution
- Control Unit Design

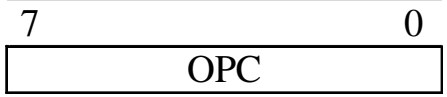




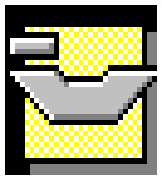
Instructions

Chosen Instructions Set

Type 1 Format

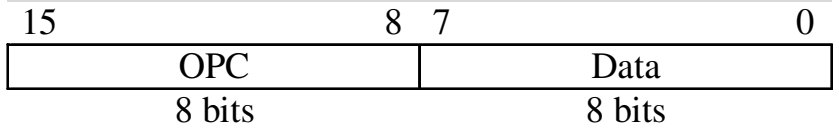


Operation	Syntax	Description	Op. Code
Addition	ADD r1	$A \leftarrow A + r1$	30h, 31h, 32h, 33h, 45h
Subtraction	SUB r1	$A \leftarrow A - r1$	18h, 19h, 1Ah, 1Bh, 46h
And	ANA r1	$A \leftarrow A \; and \; r1$	20h, 21h, 22h, 23h, 48h
Or	ORA r1	$A \leftarrow A \; or \; r1$	24h, 25h, 26h, 27h, 49h



Chosen Intructions Set

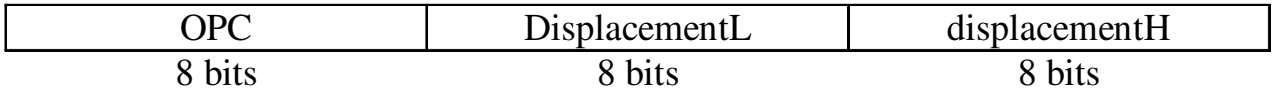
Type 2 Format



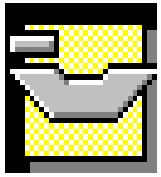
Operation	Syntax	Description	Op. Code
Immediate Addition	ADI data	$A \leftarrow A + data$	35h
Immediate Subtract.	SUI data	$A \leftarrow A - date$	36h
Immediate And	ANI data	$A \leftarrow A\ and\ date$	68h
Immediate Or	ORI data	$A \leftarrow A\ or\ date$	69h

Chosen Instructions Set

Type 3 Format



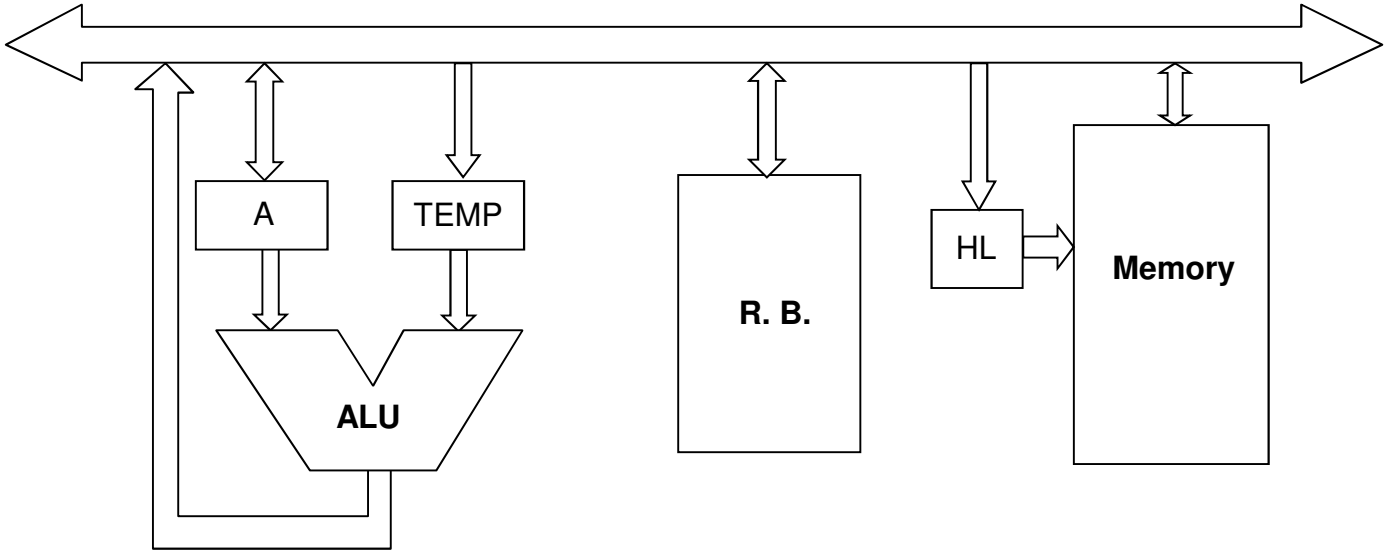
Operation	Syntax	Description	Op. Code
Load	LDA addr	$A \leftarrow M(addr)$	70h
Store	STA addr	$M(addr) \leftarrow A$	71h
Unconditional Jump	JMP addr	$PC \leftarrow addr$	74h
Jump if FZ=1	JZ addr	$If\ FZ = 1 \Rightarrow PC \leftarrow addr$	72h

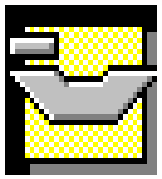


Buses use

Data path with a single bus

- 8 bits single bus for data
- 16 bits single bus for address
- Temporal registers are needed to free the bus

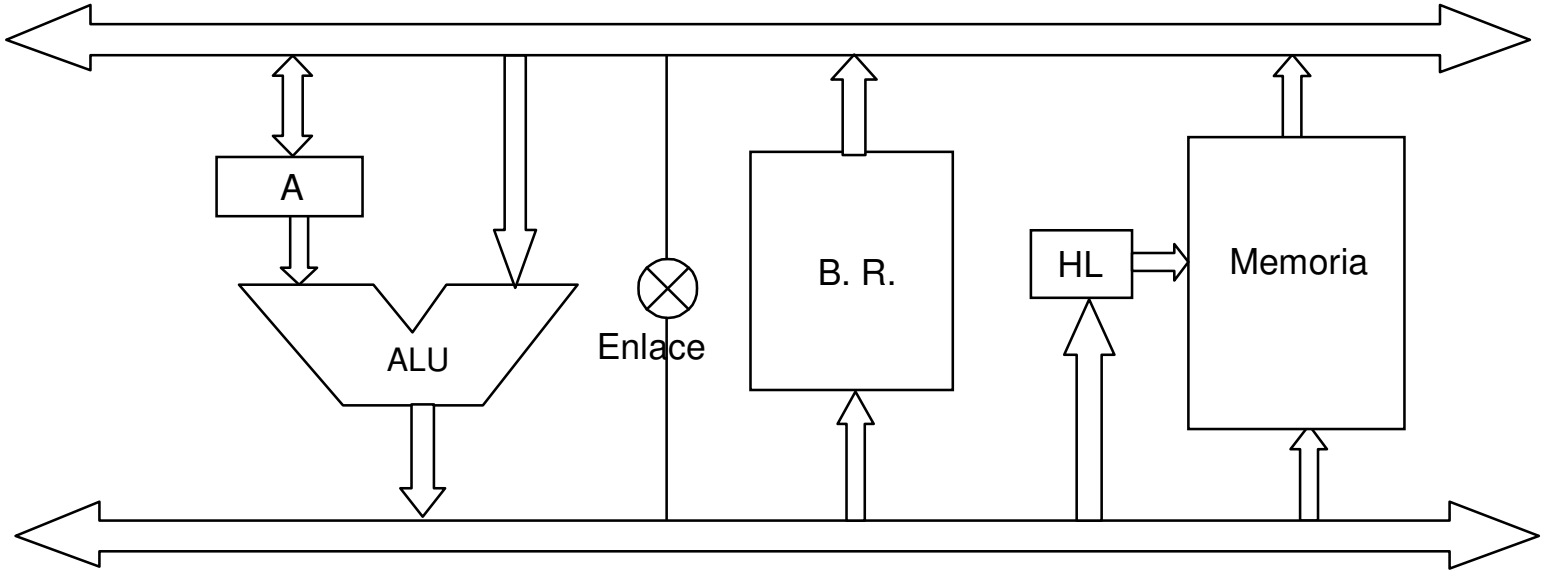


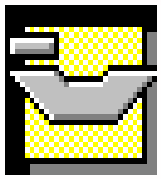


Buses use

Data path with two buses

- Two 8 bits buses
- Memory communication using HL
- Bus link

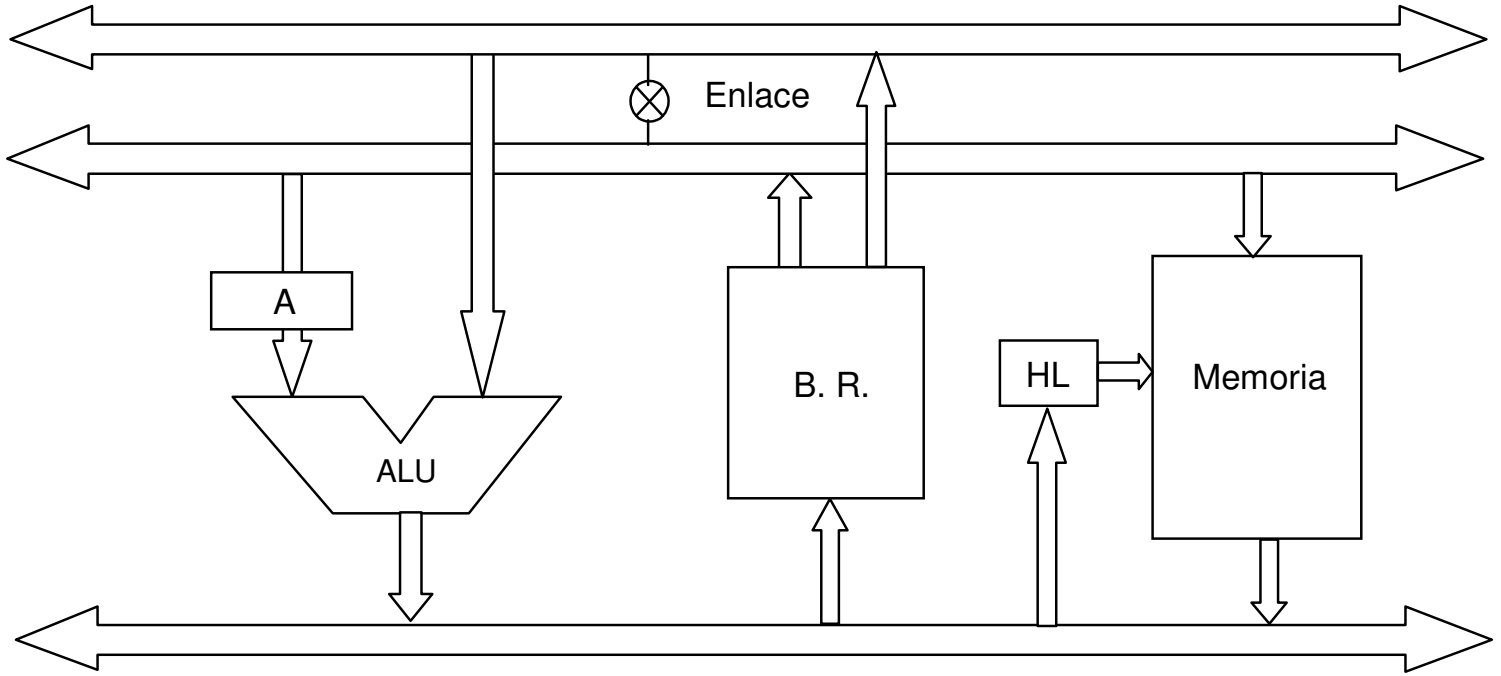




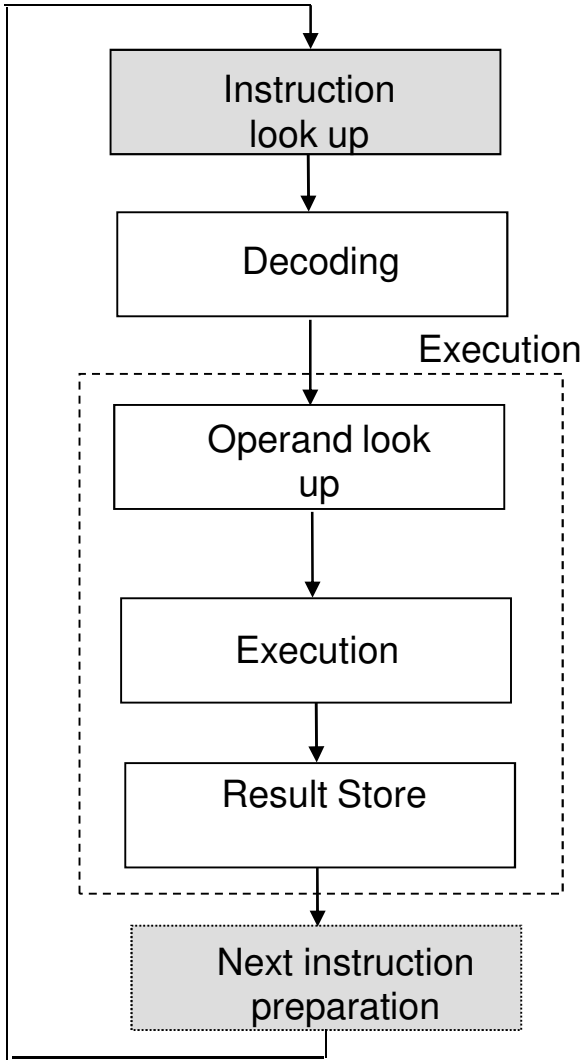
Buses use

Data path with 3 buses

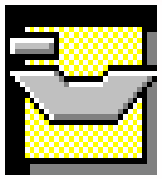
- Three 8 bits buses
- Memory communication using HL
- Bus link



Instructions Execution Stages



- Flank fired design.
- Each stage will go on for one clock cycle.
- Duration for a clock cycle will depend on the lowest action's duration.
- A read and a write operation in the register bank can not be done in the same clock cycle.

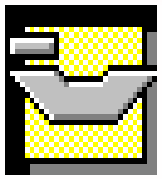


Execution Stages

Considerations

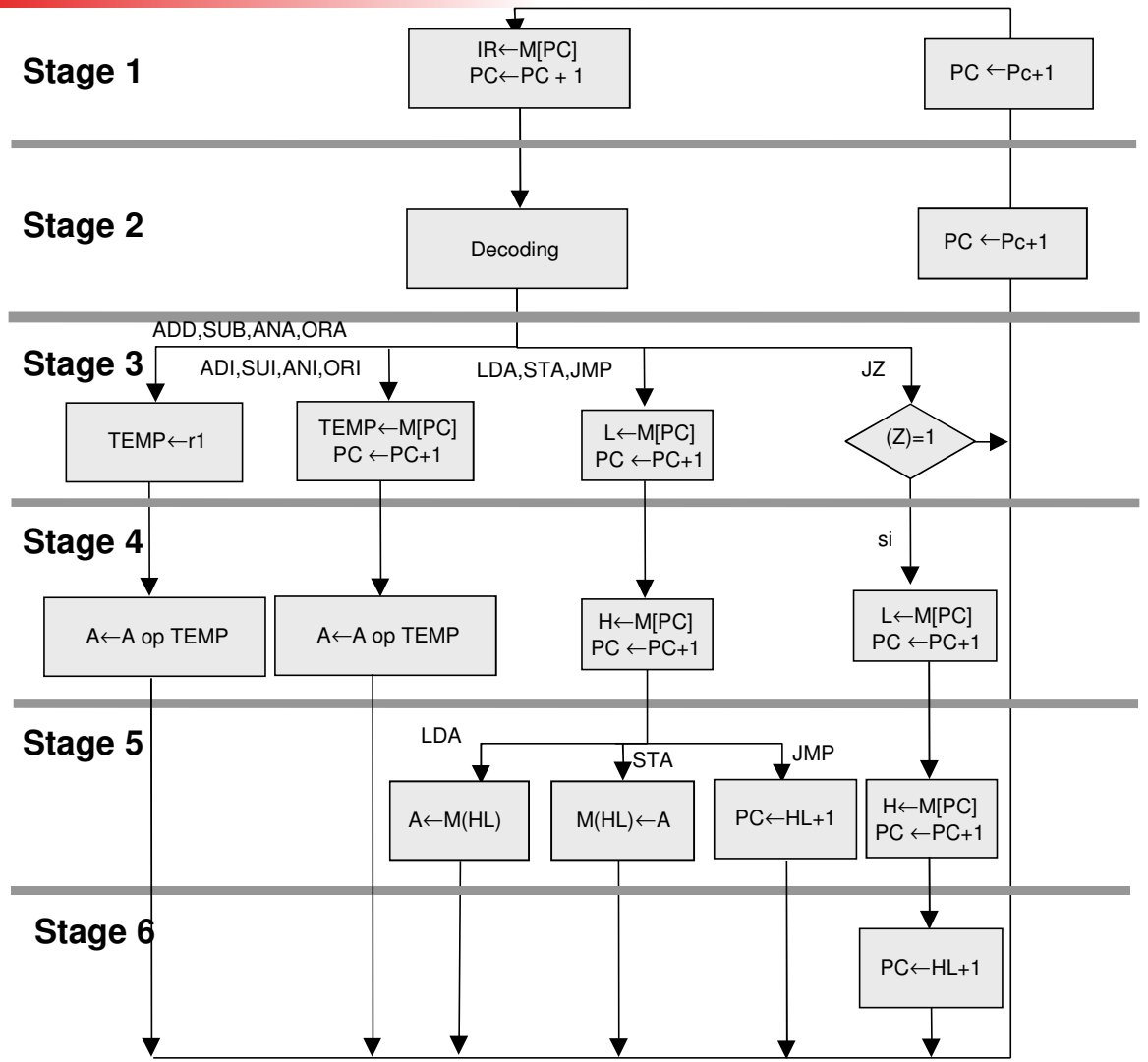
- Basic functions of MaNoTas:
  - Access to Registers Bank
  - Access to Memory
  - ALU Operations
- Assumptions:
  - The time for these functions is equal to a clock cycle.
  - Others elements' cost is zero.
- Actions bound to a stage are done in parallel.
- Actions bound to successive stages are done serially.

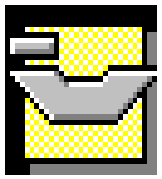
departamento de tecnología informática y computación
UNIVERSIDAD DE ALICANTE



Execution Stages

Stages

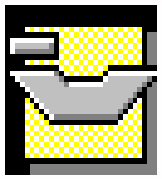




Execution Stages

Stages Establishment (1)

- **Stage1: Instruction look up**
  - IR ← M[PC]
  - PC ← PC + 1
- **Stage 2. Decoding**
  - This two stages are common to all instructions

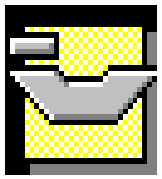


Execution Stages

Stages Establishment (2)

- **Stage 3: Operands obtaining and code evaluation for the Z condition.**
  - **Case I.** Arithmetic-Logic Instructions
    - Direct to register addressing mode
      - TEMP ← r1
    - Immediate addressing mode
      - TEMP ← M[PC] (TEMP ← data)
      - PC ← PC +1
  - **Case II.** Reference to memory instructions and unconditional jump
      - L ← M[PC] (L ← AddrL)
      - PC ← PC +1
  - **Case III.** Conditional jump instruction
      - Z?

departamento de tecnología informática y computación
UNIVERSIDAD DE ALICANTE

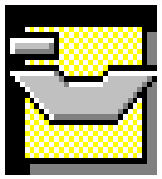


Execution Stages

Stages Establishment (3)

- **Stage 4: Operands obtaining, execution and end of Arithmetic-Logic instructions.**
  - **Case I.** Arithmetic-Logic Instructions
    - A ← A op TEMP
  - **Case II.** Reference to memory instructions and unconditional jump
    - H ← M[PC] (H ← AddrH)
    - PC ← PC +1
  - **Case III.** Conditional jump instruction
    - L ← M[PC] (L ← AddrL)
    - PC ← PC +1

departamento de tecnología informática y computación
UNIVERSIDAD DE ALICANTE

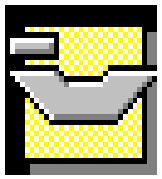


Execution Stages

Stages Establishment (4)

- **Stage 5: End of access to memory instructions and unconditional jump and operand obtaining.**
  - **Case I.** Reference to memory instructions
    - Load instruction
      - A ← M[HL]
    - Store instruction
      - M[HL] ← A
  - **Case II.** Unconditional jump instruction
      - PC ← HL +1
  - **Case III.** Conditional jump instruction
      - H ← M[PC] (H ← AddrH)
      - PC ← PC +1

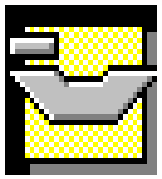
departamento de tecnología informática y computación
UNIVERSIDAD DE ALICANTE



Execution Stages

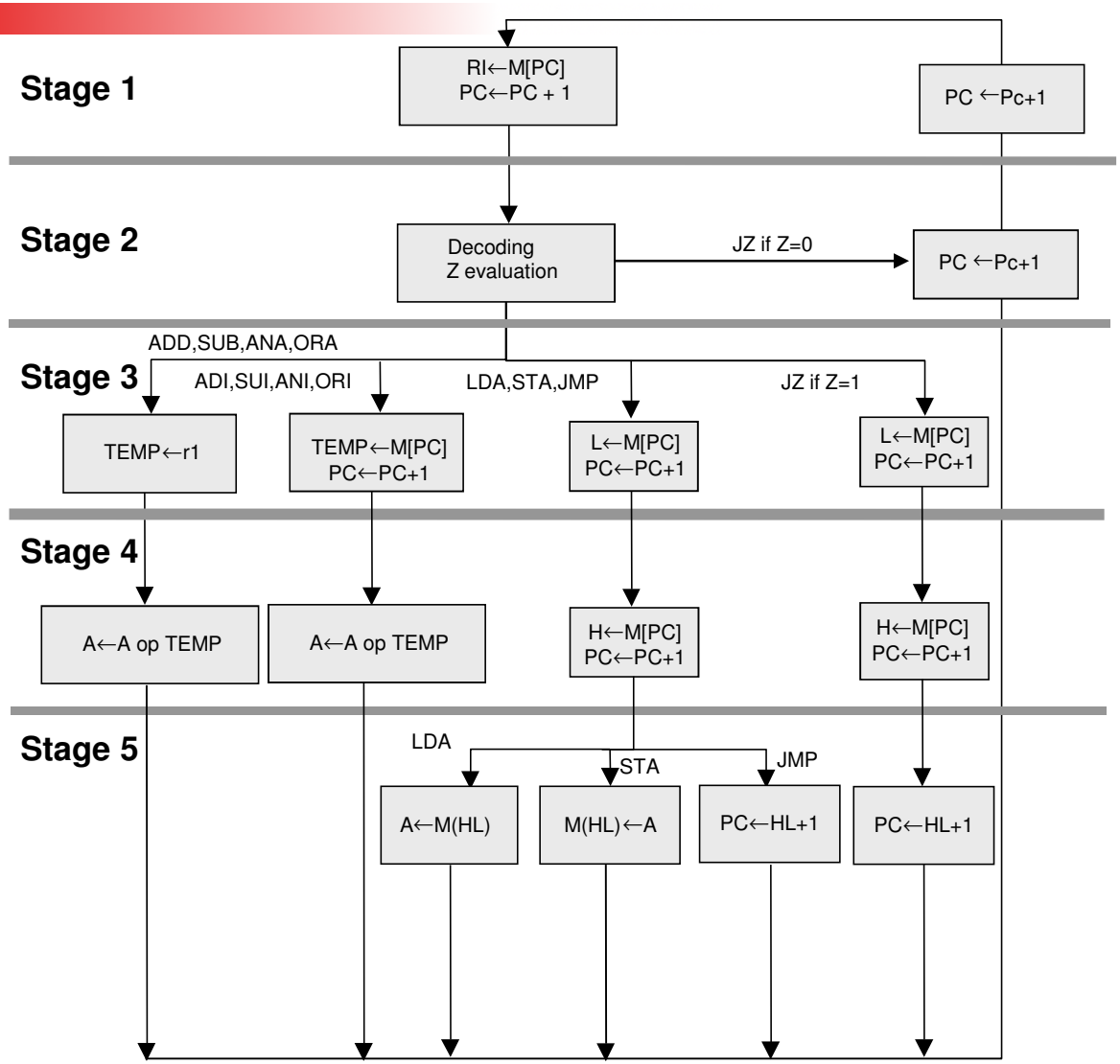
Stages Establishment (5)

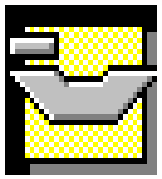
- **Stage 6: End of unconditional jump instruction.**
  - PC ← HL +1



Execution Stages

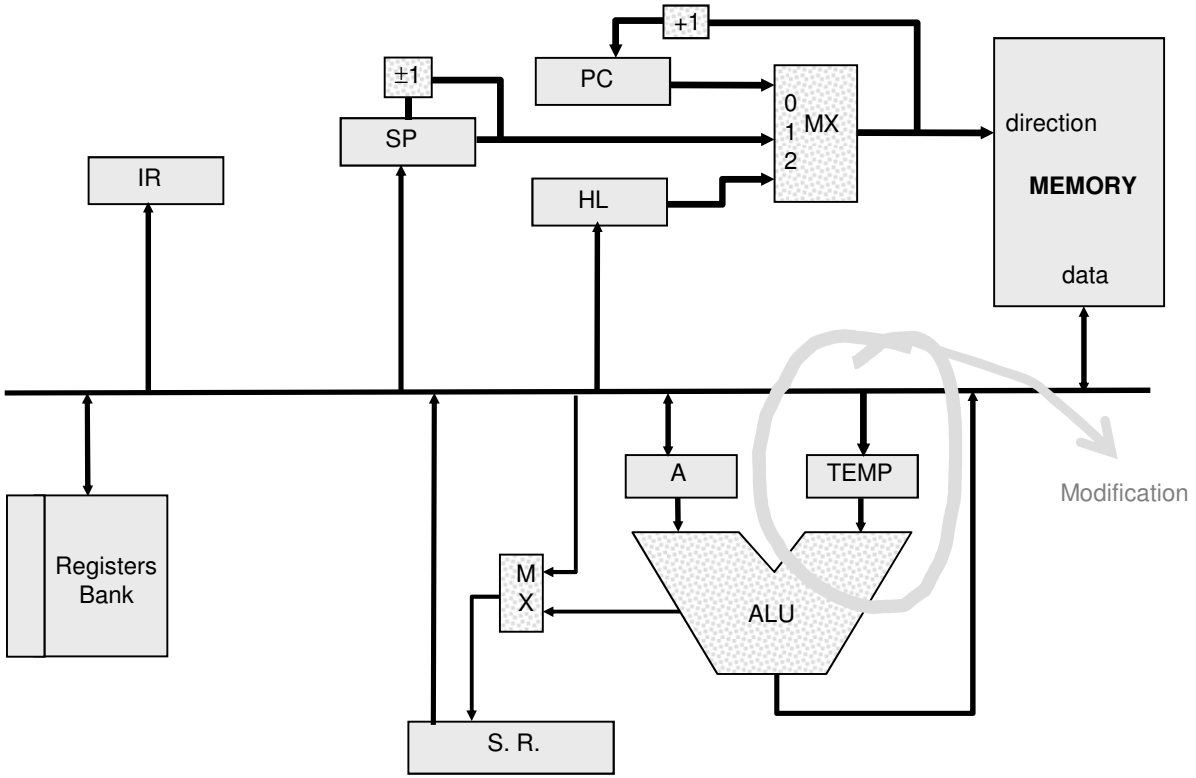
Final stages diagram

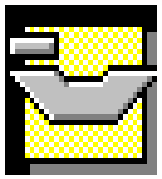




Execution Stages

Data path





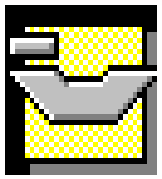
Control Unit

Identification of the control signals(1)

- Output control signals for the registers bank

Signal	Description
	REGISTERS BANK CONTROL
SELreg1 SELreg0	This two signals select the register which should be accessed: B, C, D or E.
Lreg	If it is set (value 1), allows the data that is in the bus to be stored in the selected register.
Sreg	If it is set (value 1), allows the data that is in the selected register to appears in the bus.

departamento de tecnología informática y computación
UNIVERSIDAD DE ALICANTE



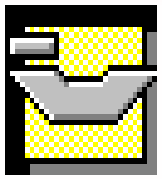
Control Unit

Identification of the control signals (2)

- Output control signals for the memory and the ALU

Signal	Description
	MEMORY CONTROL
Lmem	Memory load signal. If its value is 1, the data stored in the address pointed by the address bus is placed in the data bus.
Smem	Memory store signal. If its value is 1, the data present in the data bus is stored in the address pointed by the address bus.
SAddr2 SAddr1	These signals contol the multiplexor that selects the data to be placed in the address bus for access the memory. Depending of its value, the memory address that will be accessed would be set by the PC, SP or HL register.
	ARITHMETIC LOGIC UNIT CONTROL
ALU2 ALU1 ALU0	These three control signals set the operation that will perform the ALU: addition, subtraction, and, or, xor, increment and decrement.
Salu	If this signal is set, the ALU result will be placed in the data bus.

departamento de tecnología informática y computación
UNIVERSIDAD DE ALICANTE

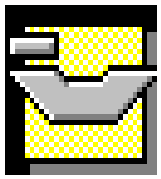


Control Unit

Identification of the control signals (3)

- Output control signals for the status register

Signal	Description
	STATUS REGISTER CONTROL
LF	If the signal is set, loads the data in the status register. The data can come from the ALU or from the accumulator register.
SF	If the signal is set, stores in the data bus the data stored in the status register.
SelO	It controls the multiplexor that selects the data to write in the status register as the Overflow flag: bit 2 from the accumulator register or the overflow flag coming from the ALU.
SelC	It controls the multiplexor that selects the data to write in the status register as the Carry flag: bit 1 from the accumulator register or the carry flag coming from the ALU.
SelZ	It controls the multiplexor that selects the data to write in the status register as the Zero flag: bit 0 from the accumulator register or the zero flag coming from the ALU.

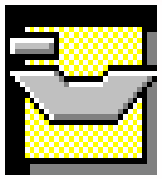


Control Unit

Identification of the control signals(4)

- Output control signals for the registers and input control signals.

Signal	Description
	REGISTERS CONTROL
Lir	If set, the data present in the data bus is stored in the instruction register.
Lpc	If set, the data present in the input of the program counter is loaded.
LspL	If set, the data present in the data bus is loaded in the low order byte of the SP register.
LspH	If set, the data present in the data bus is loaded in the high order byte of the SP register.
Isp	If set, the content of the SP register is increased by 1.
Dsp	If set, the content of the SP register is decreased by 1.
LaddrL	If set, the data present in the data bus is loaded in the low order byte of the HL register.
LaddrH	If set, the data present in the data bus is loaded in the high order byte of the HL register.
Lac	If set, the data present in the data bus is loaded in the accumulator register.
Sac	If set, the data in the accumulator register is placed in the data bus.
Ltemp	If set, the data present in the data bus is stored in the temporal register.
	CONTROL INPUTS
Z	The Z flag generated by the ALU.
OPC	Operation code, is the content of the instruction register.



Control Unit

Identification of the control signals(5)

- Control signals for the multiplexors, the register bank and the ALU.

SELreg1	SELreg0	Register
0	0	B
0	1	C
1	0	D
1	1	E

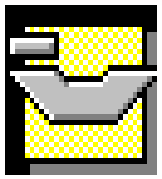
SAddr2	SAddr1	Address
0	0	PC
0	1	SP
1	0	HL
1	1	Unused

ALU2	ALU1	ALU0	Operation
0	0	0	Addition
0	0	1	Subtraction
0	1	0	And
0	1	1	Or
1	0	0	Xor
1	0	1	Increment
1	1	0	Decrement
1	1	1	Unused

SelO	Action
0	O←ALU_O
1	O←A_2

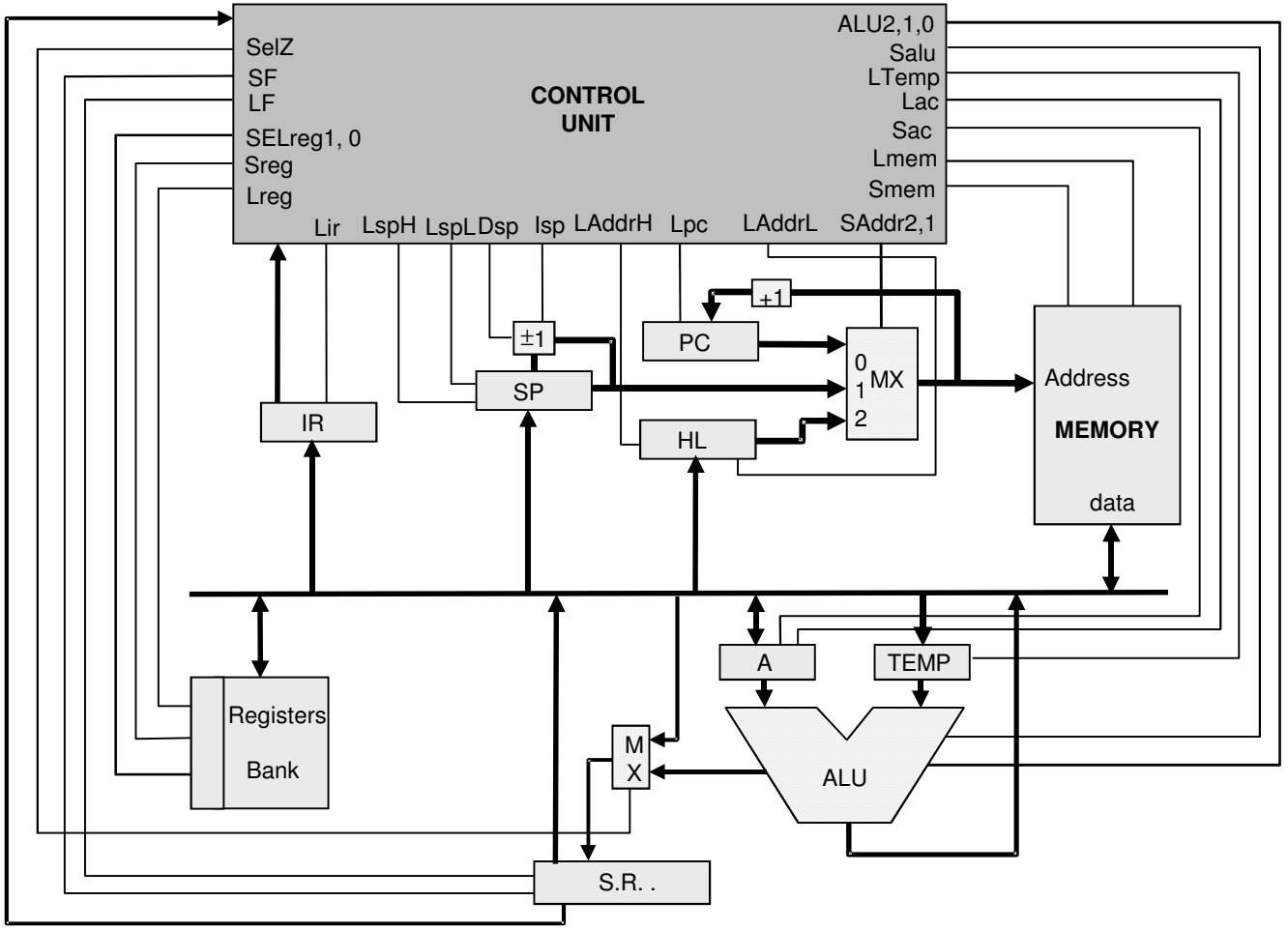
SelC	Action
0	C←ALU_C
1	C←A_1

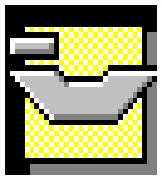
SelZ	Action
0	Z←ALU_Z
1	Z←A_0



Control Unit

Data and control path



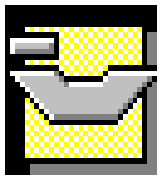


Control Unit

Control signals activation

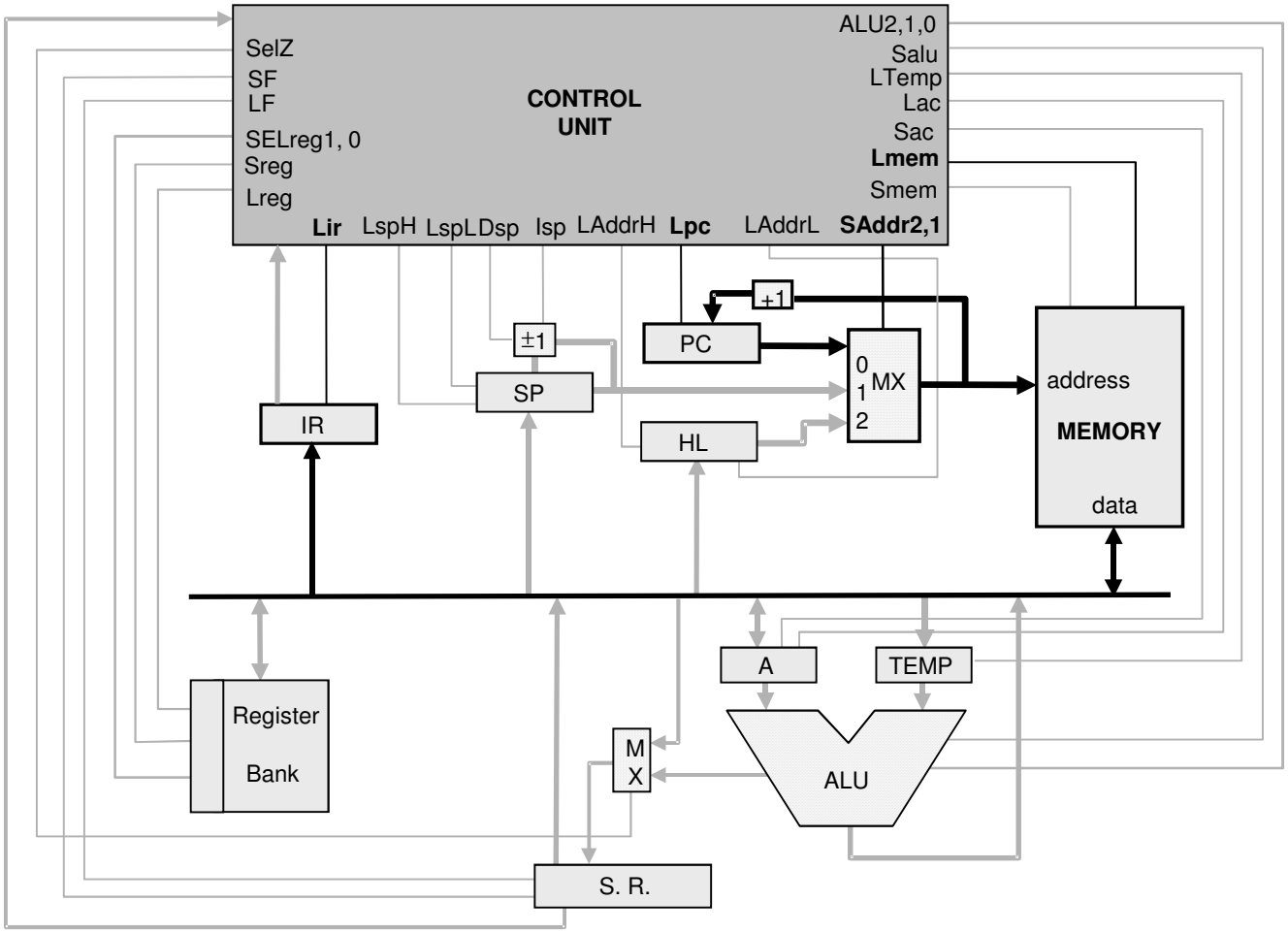
These instructions will not be considered: ADD A, SUB A, ANA A y ORA A

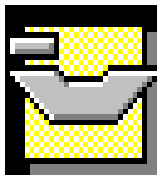
Stages	Operation	Signals activation
STAGE 1		
	RI ← M[PC] PC ← PC + 1	SAddr2, SAddr1(=00), Lmem, Lir, Lpc
STAGE 2		
	Decoding and evaluation of Z	
STAGE 3		
ADD, SUB, ANA, ORA	TEMP ← r1	SELreg1,SELreg0 (=r1), Sreg, Ltemp
ADI, SUI, ANI, ORI	TEMP ← M[PC] PC ← PC + 1	Ltemp, Lmem, SAddr2, SAddr1(=00) Lpc
LDA, STA, JMP, JZ	L ← M[PC] PC ← PC + 1	LAddrL, Lmem, SAddr2, SAddr1(=00) Lpc
STAGE 4		
Arithmetic-Logic	A ← A op TEMP	ALU2, ALU1, ALU0 (=operation), Salu, Lac
Transfer and jump	H ← M[PC] PC ← PC + 1	LAddrH, Lmem, SAddr2, SAddr1(=00) Lpc
STAGE 5		
LDA	A ← M(HL)	SAddr2,SAddr1 (=10), Lmem,Lac
STA	M(HL) ← A	SAddr2,SAddr1 (=10), Smem, Sac
JMP, JZ	PC ← HL + 1	SAddr2,SAddr1 (=10), Lpc



Control Unit

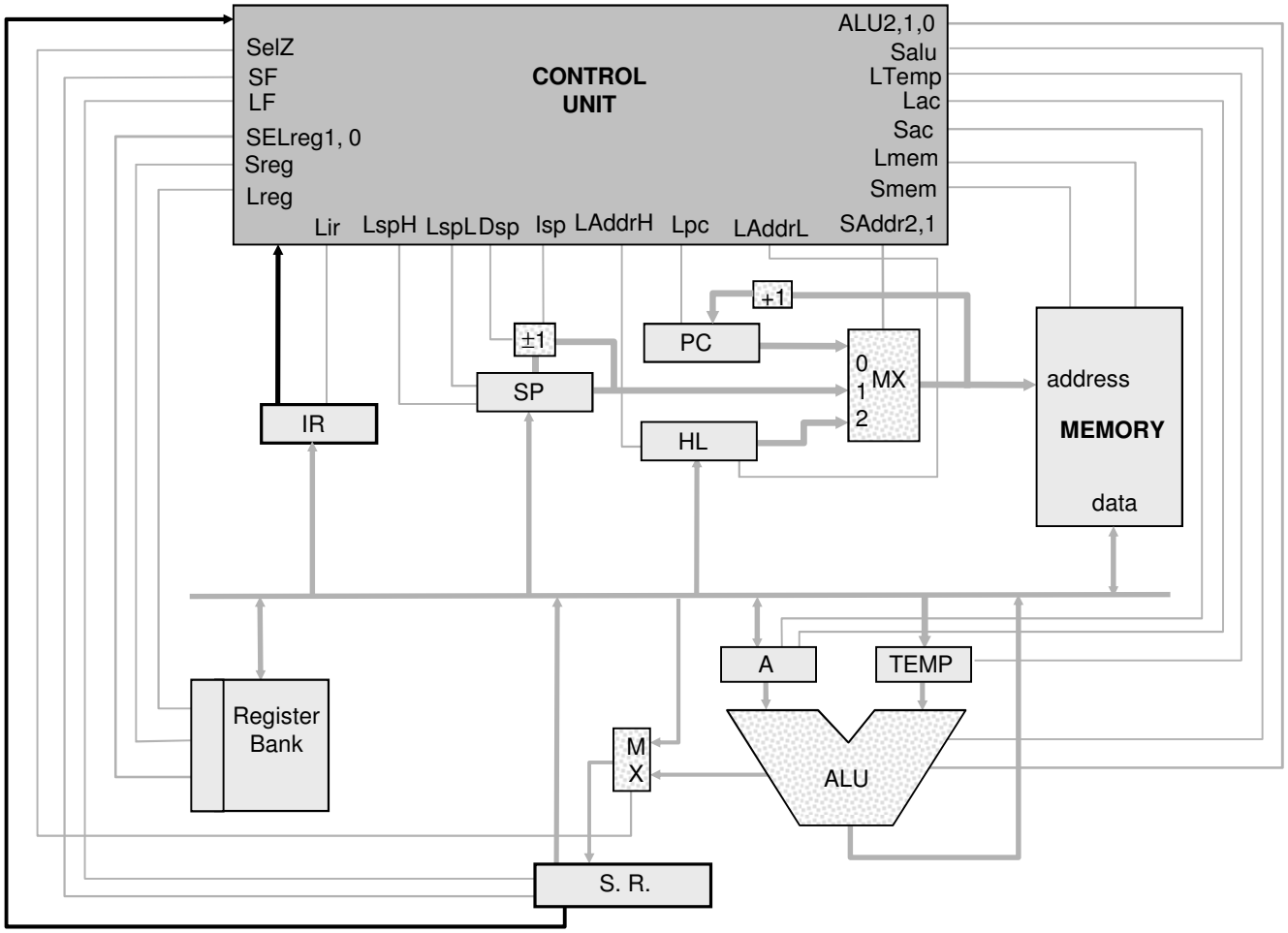
Stage 1 of the execution of LDA

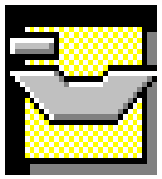




Control Unit

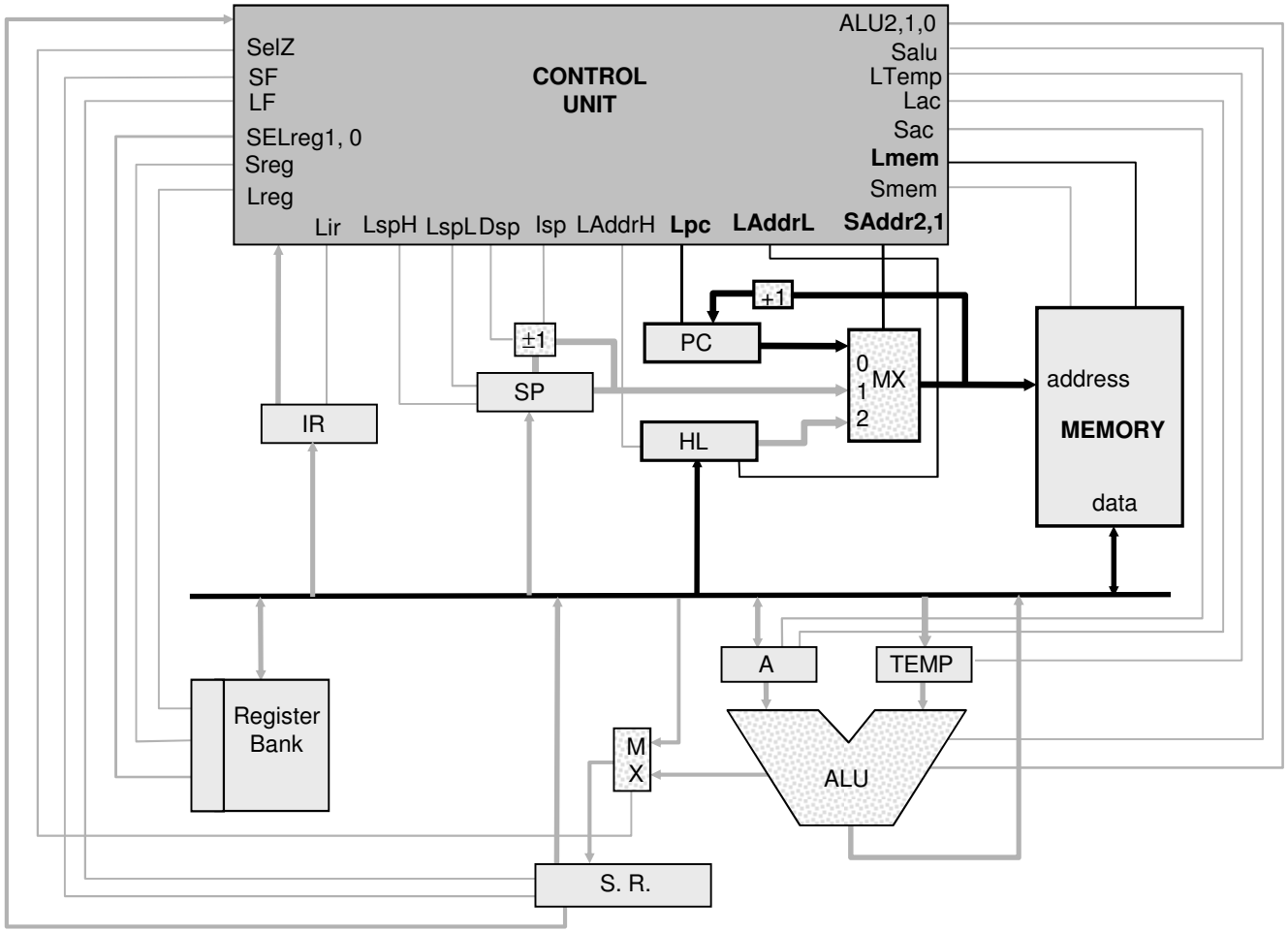
Stage 2 of the execution of LDA

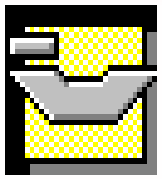




Control Unit

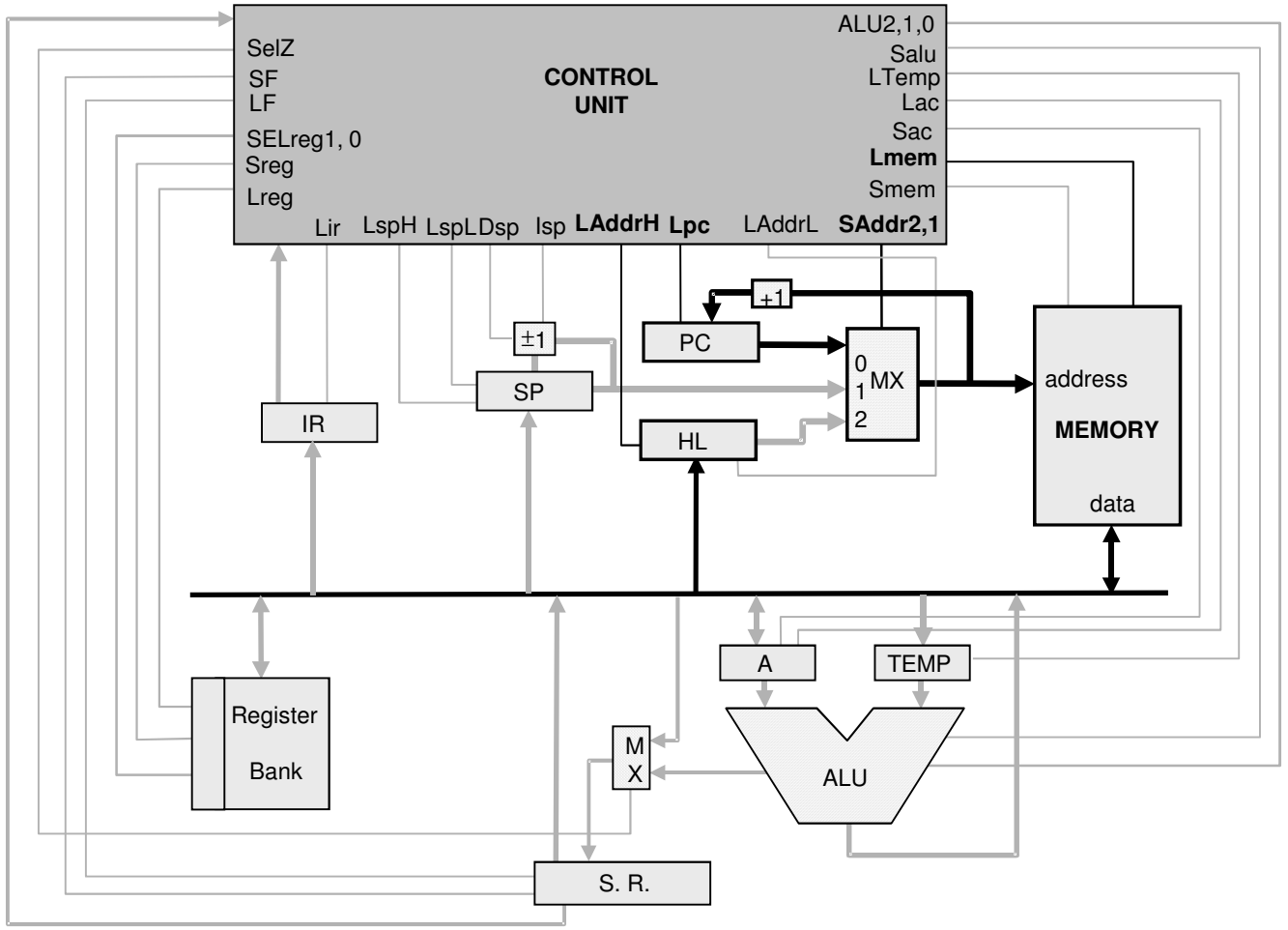
Stage 3 of the execution of LDA

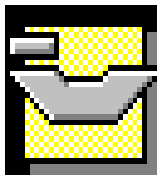




Control Unit

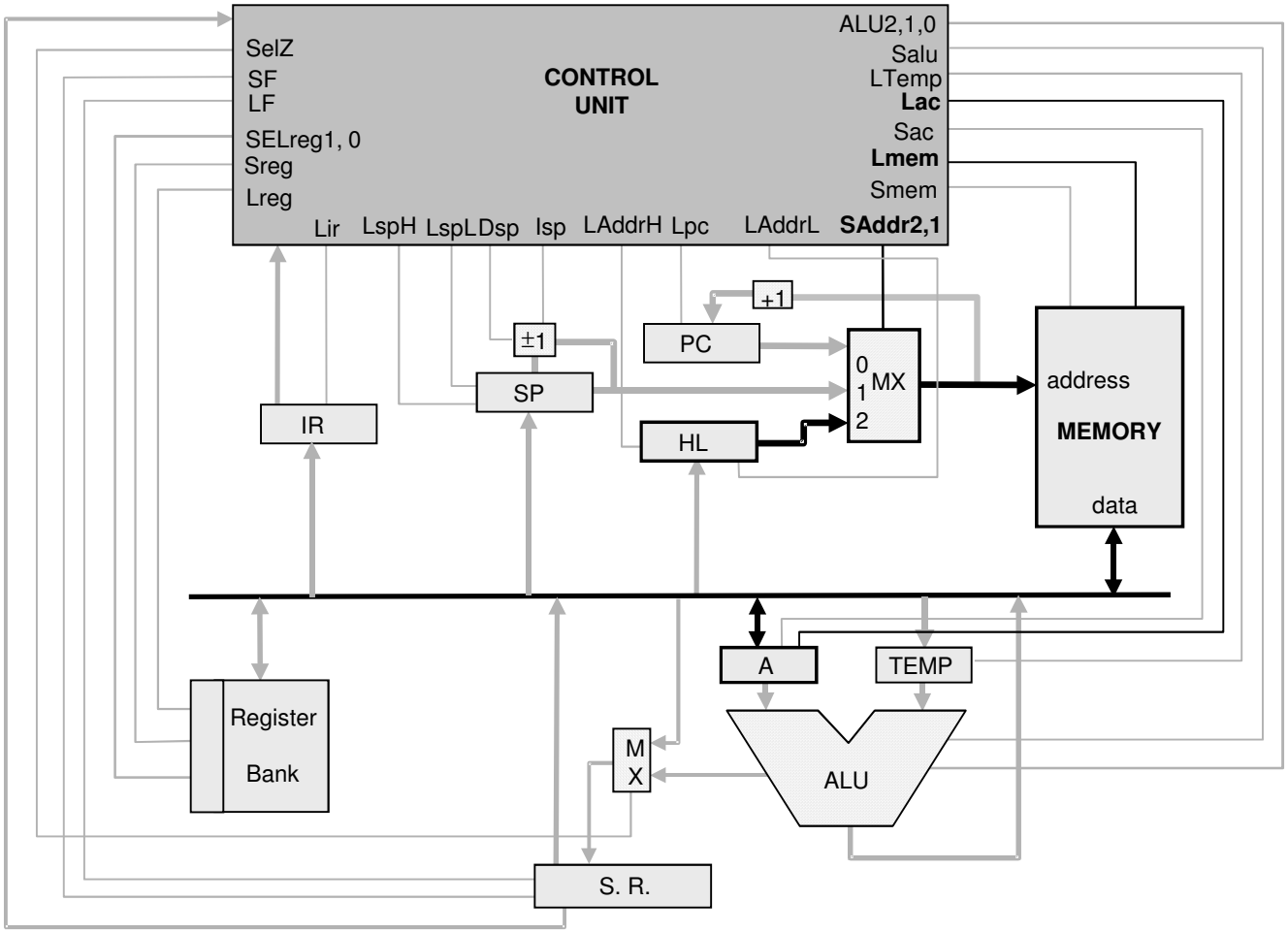
Stage 4 of the execution of LDA

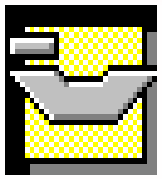




Control Unit

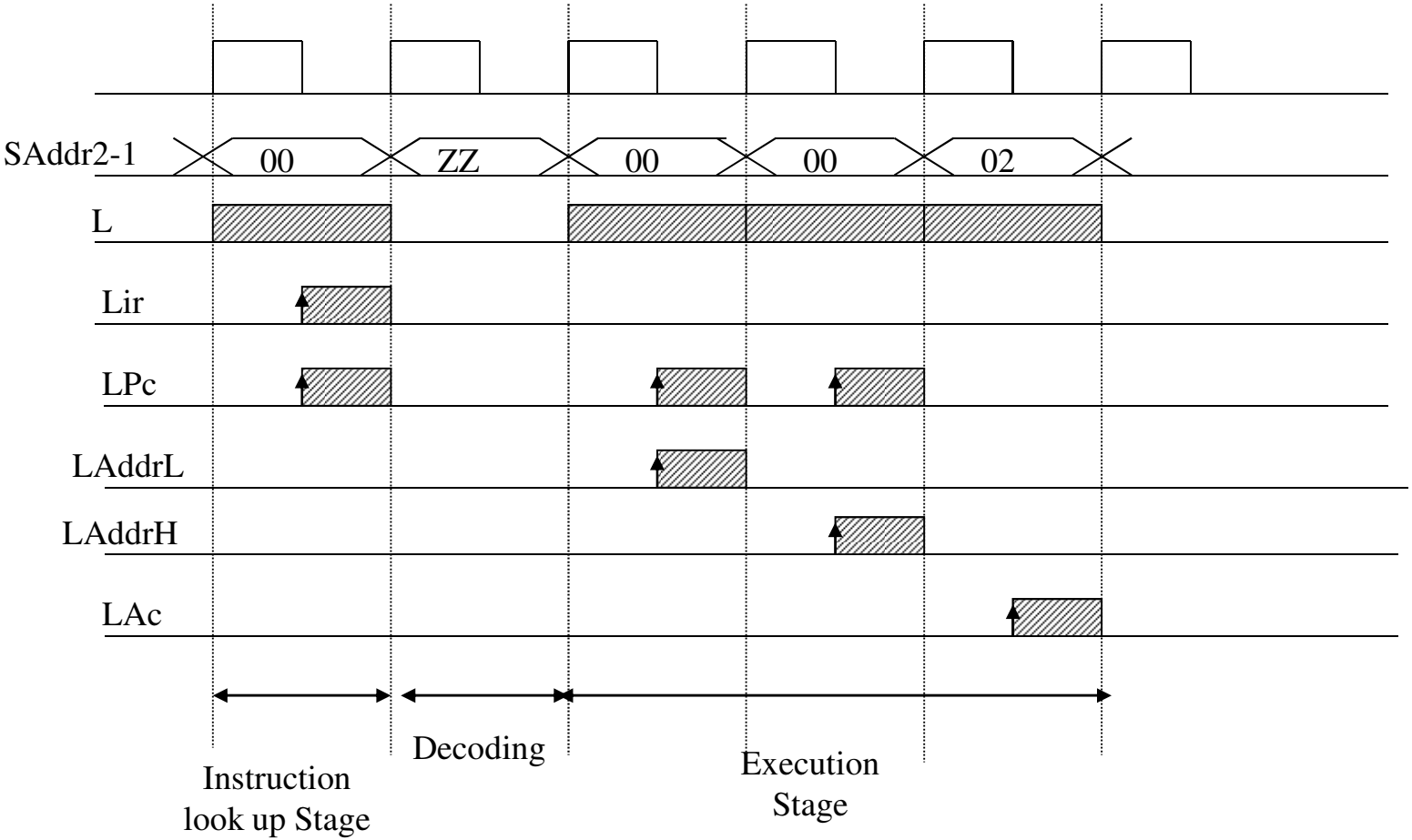
Stage 5 of the execution of LDA

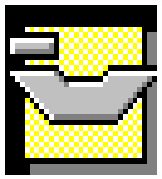




Control Unit

LDA addr



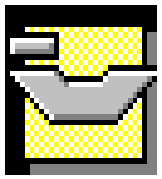


Control Unit Design

Control Unit Design

- Wired Control.
  - Implemented with Hardware, therefore very fast.
  - It is not flexible: a posterior modification implies to change the whole circuit.
- Microprogrammed Control.
  - Programmed representation for the control.
  - Slower, as the control memory must be accessed.
  - Flexible, it allows posterior modifications with out change the whole circuit.
- Two methods to design a wired control unit
  - State table method
  - Sequence counter method

departamento de tecnología informática y computación
UNIVERSIDAD DE ALICANTE



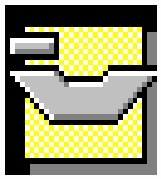
Wired Control Unit

State table method

- Based on a finite state machine.
- A finite state machine consists:
  - An internal memory that holds the state and
  - Two combinatories functions:
    - The next state function
    - The output function

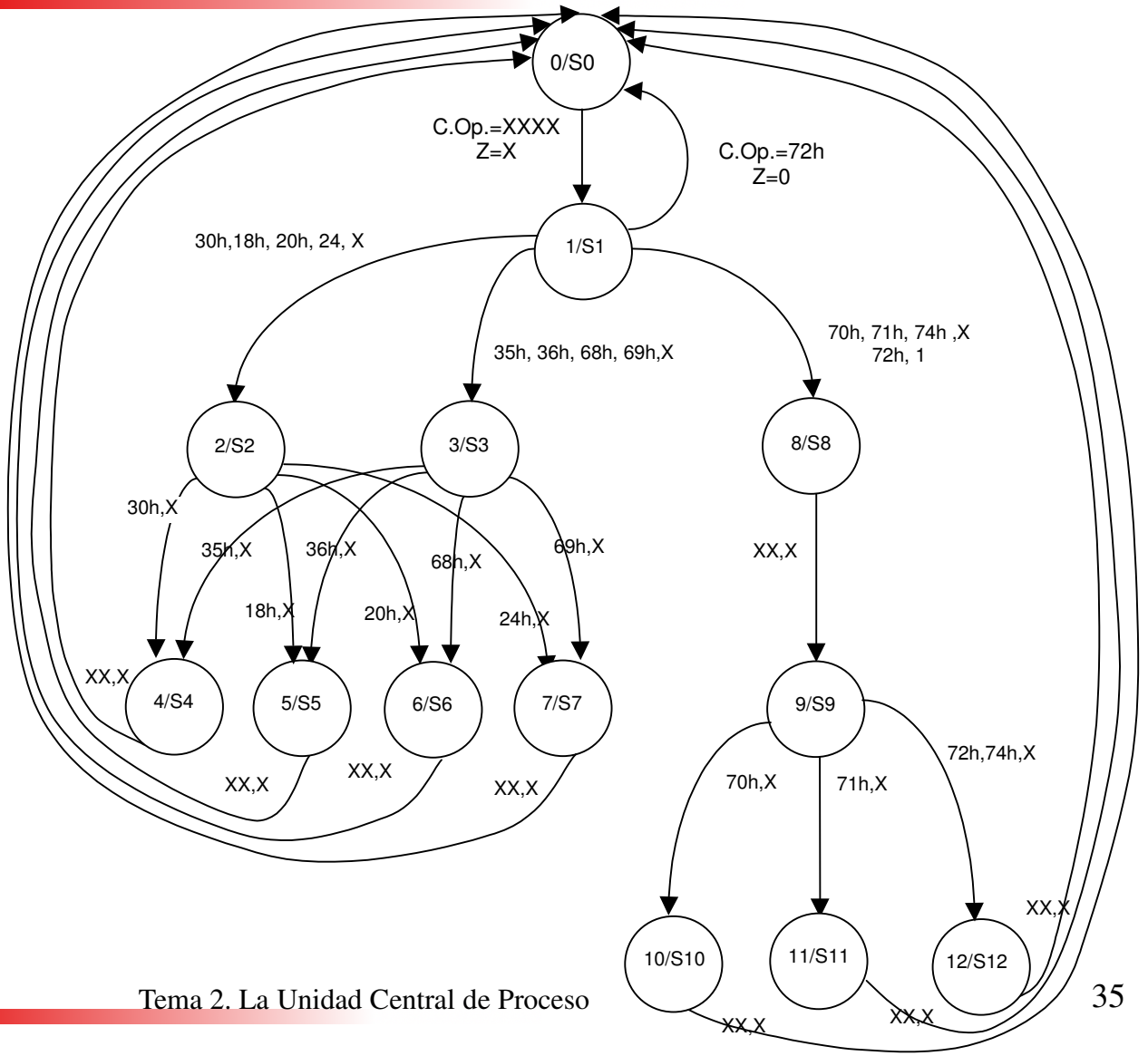
    - Each state corresponds to a clock cycle and contains the operations to be done on that cycle.
    - The next state function is a combinatory function that using its inputs and the current state determines the next state.
    - The output function produced the set of control signals from its inputs and the current state.

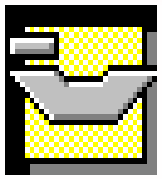
departamento de tecnología informática y computación
UNIVERSIDAD DE ALICANTE



State
Table
Method

States graph





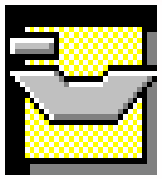
State Table Method

Output function

As arithmetic-logic instructions, we only consider: Add B, SUB B, ANA B y ORA B

Output function	States												
Control signals	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**
Lir	1	0	0	0	0	0	0	0	0	0	0	0	0
SAddr2,1	00	XX	XX	00	XX	XX	XX	XX	00	00	10	10	10
Lmem	1	0	0	1	0	0	0	0	1	1	1	0	0
Smem	0	0	0	0	0	0	0	0	0	0	0	1	0
Lpc	1	0	0	1	0	0	0	0	1	1	0	0	1
LspL	0	0	0	0	0	0	0	0	0	0	0	0	0
LspH	0	0	0	0	0	0	0	0	0	0	0	0	0
Isp	0	0	0	0	0	0	0	0	0	0	0	0	0
Dsp	0	0	0	0	0	0	0	0	0	0	0	0	0
LaddrL	0	0	0	0	0	0	0	0	1	0	0	0	0
LaddrH	0	0	0	0	0	0	0	0	0	1	0	0	0
SELreg1,0	XX	XX	00	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX
Lreg	0	0	0	0	0	0	0	0	0	0	0	0	0
Sreg	0	0	1	0	0	0	0	0	0	0	0	0	0
Lac	0	0	0	0	1	1	1	1	0	0	1	0	0
Sac	0	0	0	0	0	0	0	0	0	0	0	1	0
Ltemp	0	0	1	1	0	0	0	0	0	0	0	0	0
ALU2,1,0	XXX	XXX	XXX	XXX	000	001	010	011	XXX	XXX	XXX	XXX	XXX
Salu	0	0	0	0	1	1	1	1	0	0	0	0	0
Sel0	X	X	X	X	0	0	0	0	X	X	X	X	X
SelC	X	X	X	X	0	0	0	0	X	X	X	X	X
SelZ	X	X	X	X	0	0	0	0	X	X	X	X	X
LF	0	0	0	0	1	1	1	1	0	0	0	0	0
SF	0	0	0	0	0	0	0	0	0	0	0	0	0

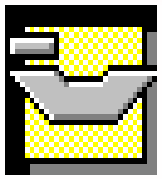
departamento de tecnología informática y computación
UNIVERSIDAD DE ALICANTE



State Table Method

Next state function

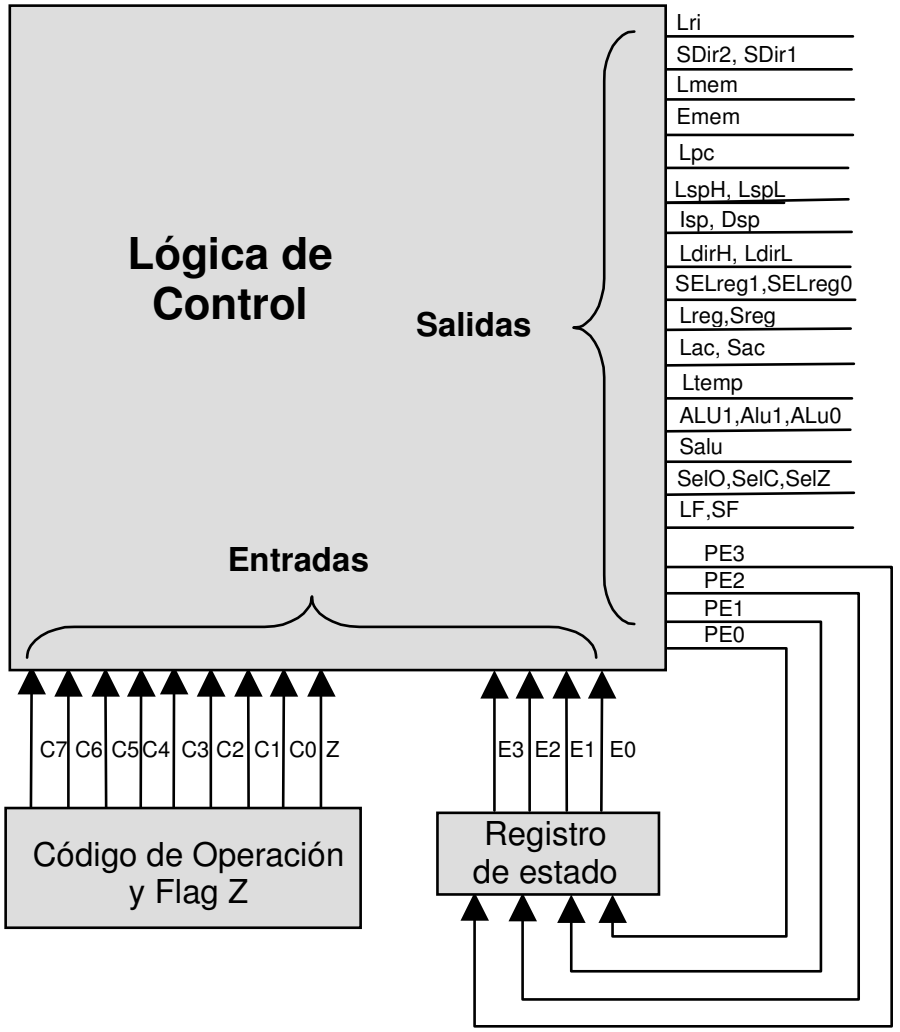
INSTRUCTION	E3	E2	E1	E0	PE3	PE2	PE1	PE0
	0	0	0	0	0	0	0	1
72h,0 (JZ)	0	0	0	1	0	0	0	0
30h,18h,20h,24h,X (ADD,SUB,ANA,ORA)	0	0	0	1	0	0	1	0
35h,36h,68h,69h,X (ADI,SUI,ANI,ORI)	0	0	0	1	0	0	1	1
70h,71h,74h,X (LDA,STA,JMP);72h,1 (JZ)	0	0	0	1	1	0	0	0
30h,X (ADD)	0	0	1	0	0	1	0	0
18h,X (SUB)	0	0	1	0	0	1	0	1
20h,X (ANA)	0	0	1	0	0	1	1	0
24h,X (ORA)	0	0	1	0	0	1	1	1
35h,X (ADI)	0	0	1	1	0	1	0	0
36h,X (SUI)	0	0	1	1	0	1	0	1
68h,X (ANI)	0	0	1	1	0	1	1	0
69h,X (ORI)	0	0	1	1	0	1	1	1
	0	1	0	0	0	0	0	0
	0	1	0	1	0	0	0	0
	0	1	1	0	0	0	0	0
	0	1	1	1	0	0	0	0
	1	0	0	0	1	0	0	1
70h,X (LDA)	1	0	0	1	1	0	1	0
71h,X (STA)	1	0	0	1	1	0	1	1
72h,74h , X (JMP, JZ)	1	0	0	1	1	1	0	0
	1	0	1	0	0	0	0	0
	1	0	1	1	0	0	0	0
	1	1	0	0	0	0	0	0

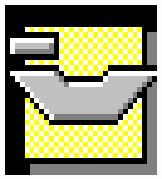


State Table Method

Control Unit Implementation

- Control unit built as a finite state machine.
- The combinatory circuit could be implemented using a ROM or a PLA.



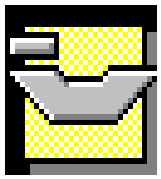


State Table Method

Control Unit Implementation

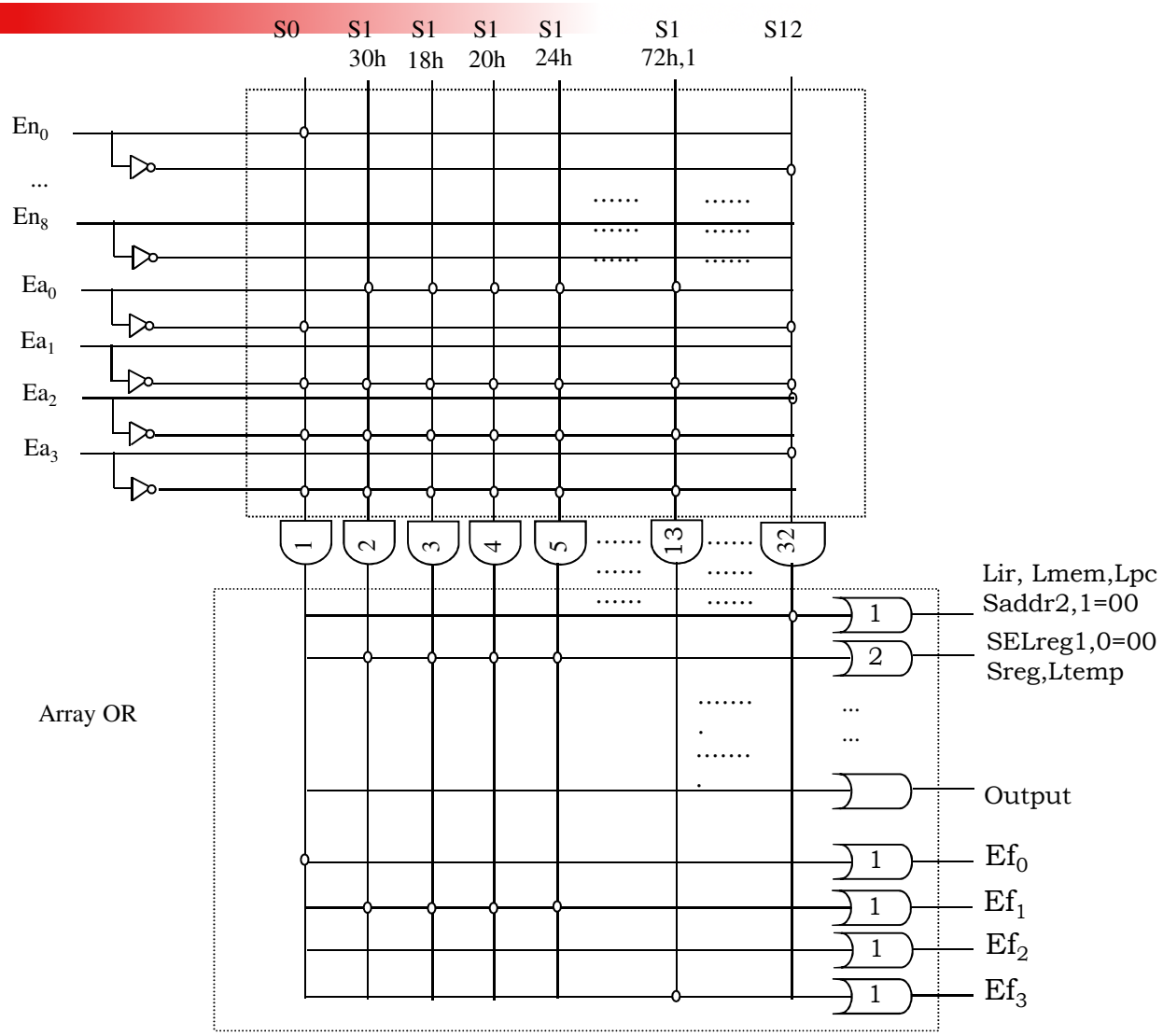
- A PLA (programmable logic array) is a general purpose programmable logic element to implement any combinatory function.
- A PLA is form by:
  - A set of inputs and outputs
  - A set of AND gates that form a set of product terms.
  - A set of OR gates, each of them form a logic sum of any product term.
  - A set of negators for the inputs.
  - Two arrays of modifiable connections (AND & OR)

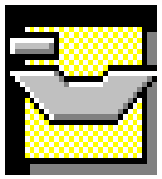
departamento de tecnología informática y computación
UNIVERSIDAD DE ALICANTE



State Table Method

Control Unit Implementation



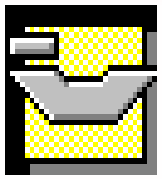


State Table Method

Control Unit Implementation

- The size of a PLA is equal to the addition of the size of the AND gates array and the size of the OR gates array.
- For MaNoTaS = (13xnumber of different product terms) + (32xnumber of sum terms).
- A PLA is more efficient than a ROM because it doesn't stores a full truth table, instead it does a minimal addition of products.
- Programming a PLA is harder than programming a ROM.

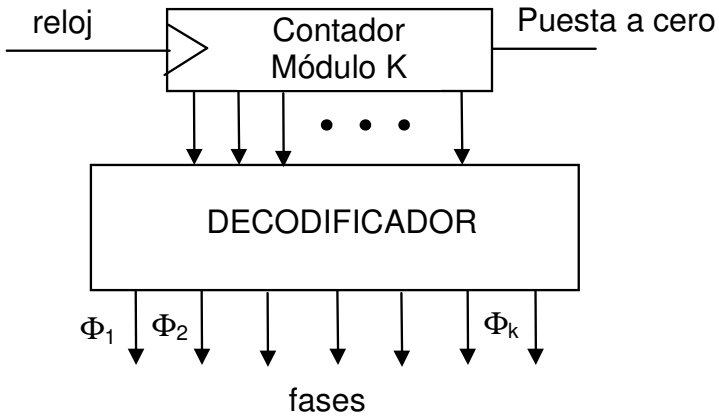
departamento de tecnología informática y computación
UNIVERSIDAD DE ALICANTE

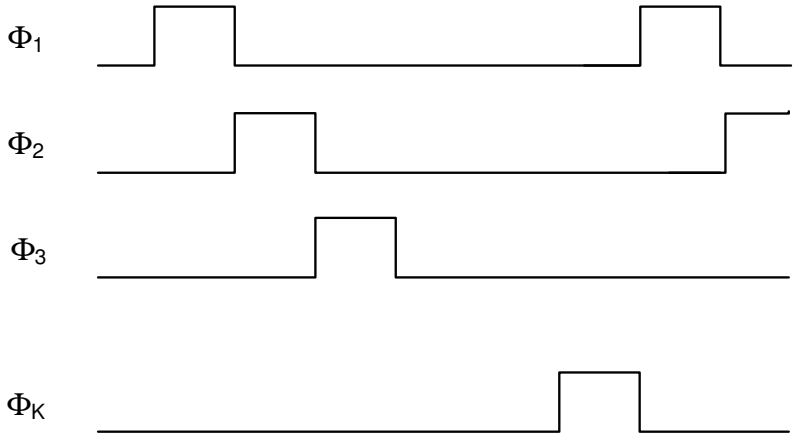


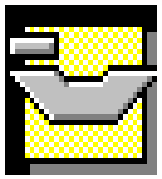
Wired Control Unit

Sequence counter method

- It is based on a module K counter which outputs are connected to a decoder generating a individual signal on each stage.
  - The counter pass throught each state on each cycle
  - The decoder generates K pulsed signals $\{\Phi_j\}$ which are the stage signals.
  - All the stages have the same duration (equal to a clock cycle)



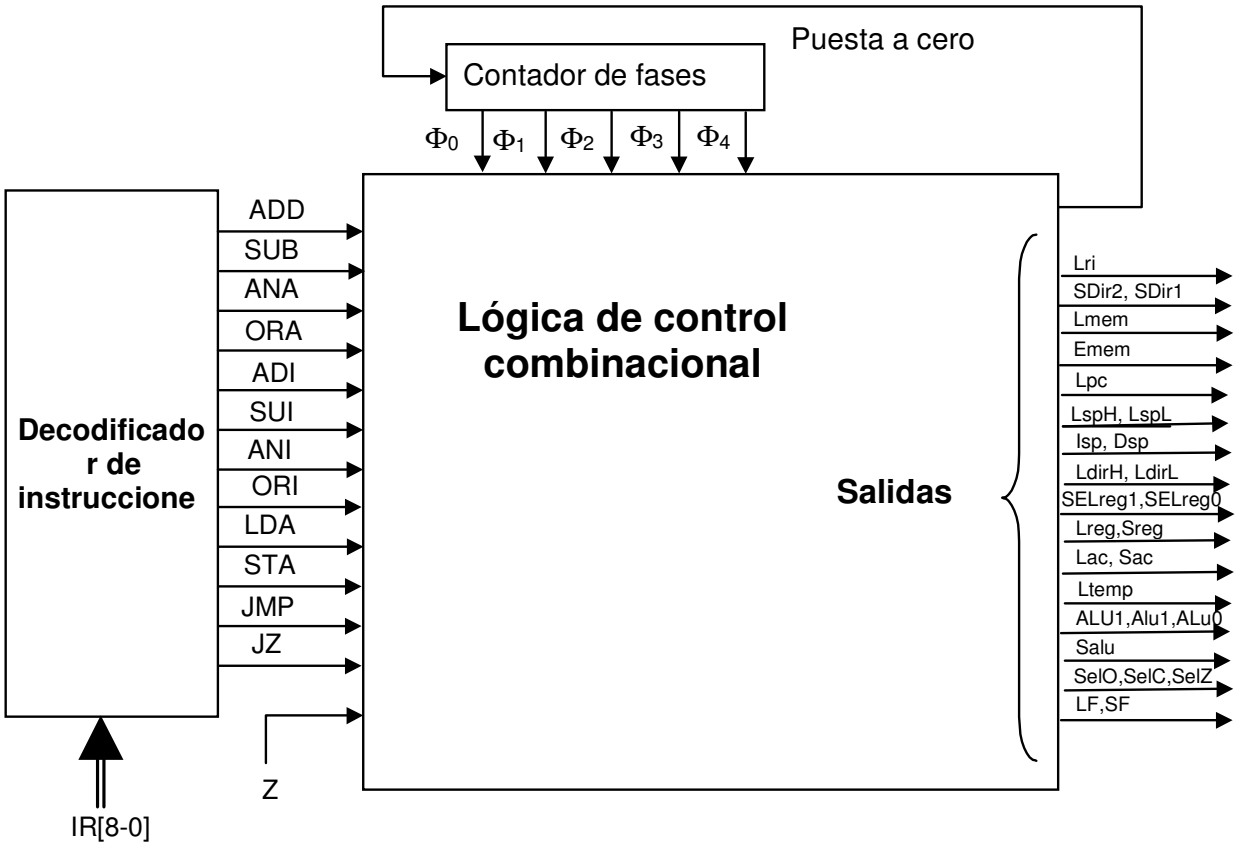


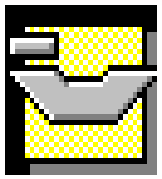


Stages Counter Method

Control Unit Implementation

- Each control signal is obtained as: $c_i = \sum_j \Phi_j \sum_m \mathrm{Im}\, S_l$



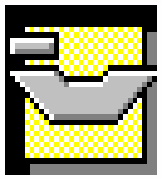


Stages
Counter
Method

Logic function examples

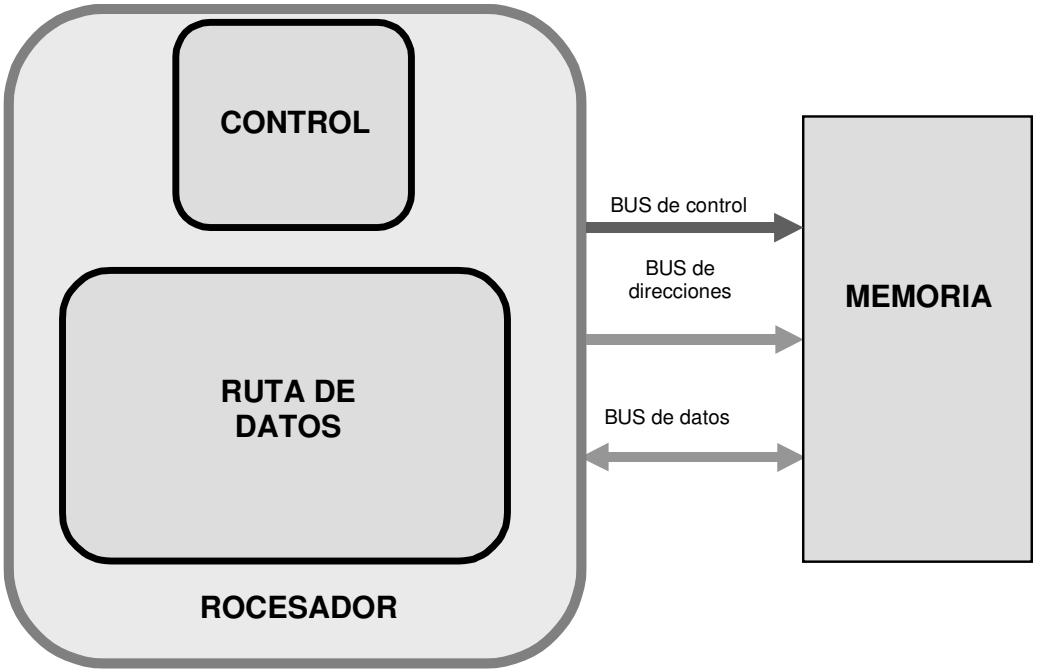
- Output logic functions for some control signals.

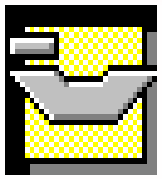
Signal	Output logic function
Zero	$\Phi 14 \cdot (JZ \cdot \overline{Z}) + \Phi 4 \cdot (add + adi) + \Phi 5 \cdot (sub + sui) + \Phi 6 \cdot (ana + ani) + \Phi 7 + \Phi 10 \cdot lda + \Phi 11 \cdot sta + \Phi 12 \cdot (jmp + JZ \cdot Z)$
Lpc	$\Phi_0 + \Phi_2 \cdot (adi + sui + ani + ori + lda + sta + jmp + jz \cdot Z)$
ALU1	$\Phi_3 \cdot (ana + ani + ora + ori)$
ALU0	$\Phi_3 \cdot (sub + sui + ora + ori)$
Smem	$\Phi_4 \cdot sta$



MaNoTas

Machine abstraction





Conclusions

- The data path structure has influence on the establishment of the stages.
- The structure of the instruction set has influence on the data path.

- The wired Control Unit is very efficient as the control signals are activated directly by the hardware.
  - It is suitable when the instructions set is not much complex.
  - However it is not flexible on posterior modifications.
- The different design methods for the control unit are all equivalents, differing only by its facility to obtain the logic functions for the control signals.