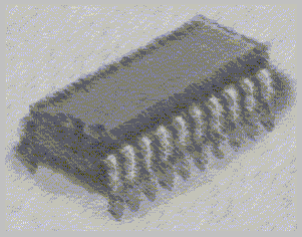
Lesson 3. The Control Unit

Computers Structure and Organization



Graduated in Computer Sciences
Graduated in Computers Engineering

Contents

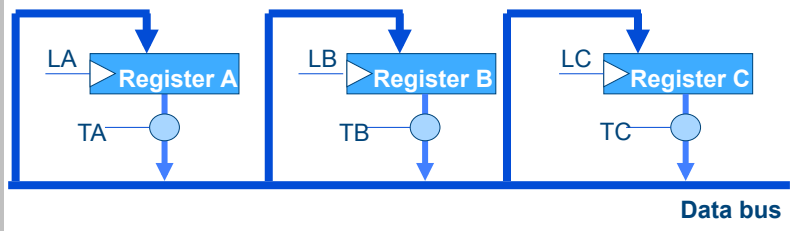
Elemental operations (I)

- **Control Unit perform following tasks:**
  - Read pointed by PC current instruction from memory (*fetch*)
  - Instruction decoding
  - Instruction execution
  - Program Counter update
- **Instruction execution is controlled by the period counter. Control Unit will generate needed signals in such periods. Control Unit get information about the instruction from:**
  - Machine code instruction
  - Flag register
  - I/O signals (interrupts, DMA, *reset*, ...)

Elemental operations (II)

- Instruction execution requires a set of steps to be performed by activating control signals. These steps are named elemental operations.
- Elemental operations are classified in:
  - **Transfer operations .** Move information from one element to another.
  - **Process operation.** Source information pass through an operator.
- **The whole elemental operations, whatever transference or whatever processes one, must start in a memory element and must also end in a memory element**

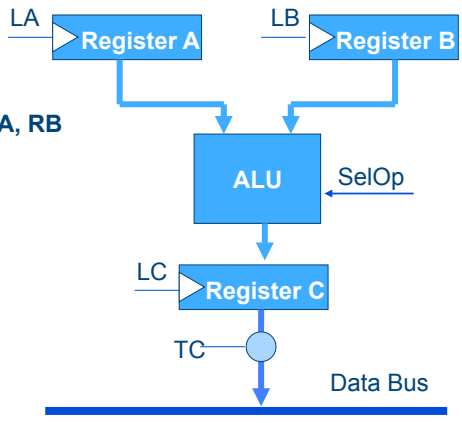
Elemental operations (III)
Transference



Transfer operation: MOV RegB, RegA

- **Pass Registro A content to Data Bus**
- **Load Data Bus content in Register B**

Elemental operations (and IV)
Process

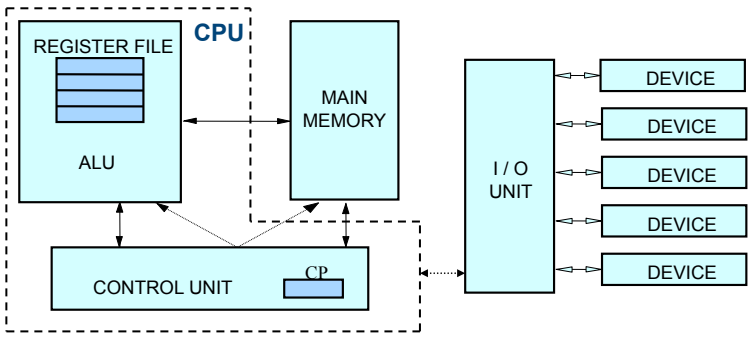


Process operation: XOR RC, RA, RB

- **Carry Register A to ALU**
- **Carry Register B to ALU**
- **Choose XOR operation**
- **Load result in Register C**

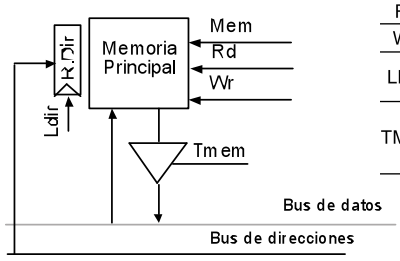
An elemental computer structure (I)

- Generated control signals by the computer control unit will be studied by using a simplified von Neuman computer architecture



An elemental computer structure (II)
Main Memory (I)

- Main memory block is built by:
  - RAM memory modules
  - Addresses register
  - Bidirectional data bus buffer

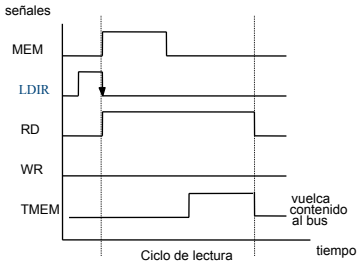


Control signals are:

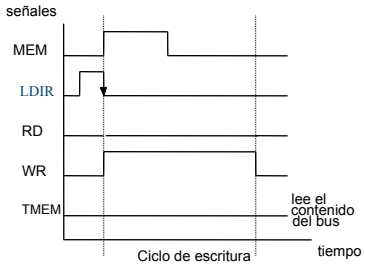
Signal	Activated by	Name	Meaning
MEM	Level	*Memory*	Start memory cycle
RD	Level	*Memory read*	Reading cycle
WR	Level	*Memory write*	Writing cycle
LDIR	Falling edge	*Address latch enable*	Load adresses register
TMEM	Level	---	When high level data pass to data bus

An elemental computer structure (III) Main Memory (and II)

- RD and WR timming depend on memory implementation
- MEM period is the same as the computer clock period



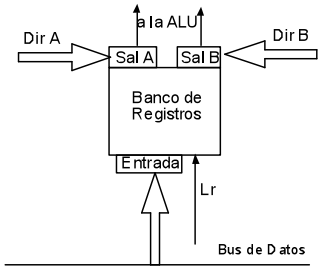
Reading cycle chronogram



Writing cycle chronogram

An elemental computer structure (IV) Register File

- Register files is built by:
  - 16 general purpose registers
  - 2 output ports that allow reading of two register at once
  - 1 input port that allows to write on a single register from the data bus



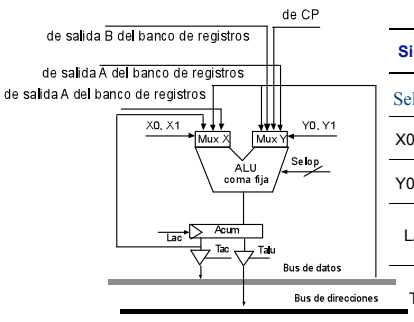
Control signal is:

Signal	Activated by	Name	Meaning
LR	Falling Edge	*Load register*	Information of the data bus is stored on DIRECCIÓN A pointed register

An elemental computer structure (V) Unidad Aritmethical-Logical Unit (I)

- ALU elements are:
  - A 4 control signal operator to choose desired operation
  - 2 4x1 multipelexor to choose operands
  - An accumulator register for storing temporal data
  - Accumulator output may be tranfered to multiplexor X, data bus or addresses bus depending on control signals

An elemental computer structure (VI) Unidad Aritmethical-Logical Unit (and II)

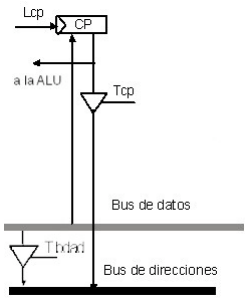


Control signals are:

Signal	Activated by	Name	Meaning
Selop	Level	*Selection*	Select operation to Be performed
X0, X1	Level	---	Choose among 3 operands
Y0, Y1	Level	---	Choose among 4 operands
LAC	Falling edge	*Load acumulator*	Loads accumulator with operator output
TAC	Level	*Data transfer*	Transfers accumulator content to data bus
TALU	Level	*Address transfer*	Transfiers accumulator content to addressses bus

An elemental computer structure (VII) Addressing Unit

- Addressses unit generates memory adressses to be accessed
- PC must be updated with the size of the current instruction. Such increment will be in data bus

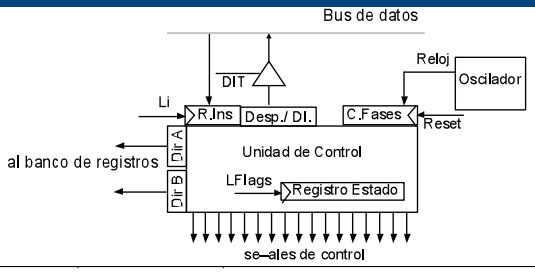


Signal	Activated by	Name	Meaning
LPC	Falling Edge	*Load program counter*	A new value is store don PC from data bus
TBDAD	Level	*Data bus transfer*	Transfers data bus contents to addresses bus
TCP	Level	*Program counter transfer*	PC content is transferred to addresses bus

An elemental computer structure (VIII) Control Unit (I)

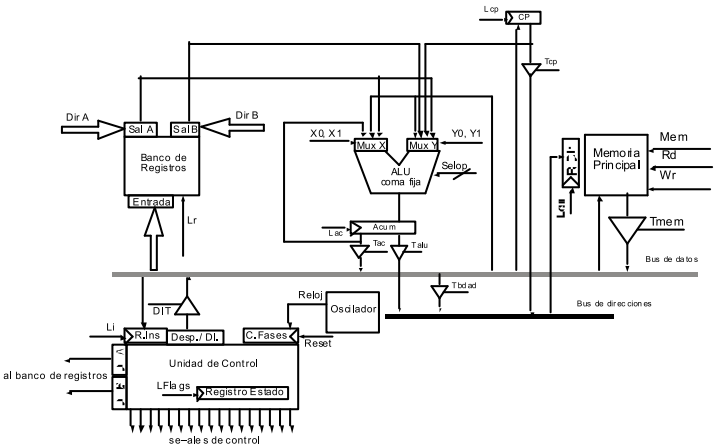
- Control unit generates the whole signals of the control bus
- Control Unit needs:
  - **Instruction register**,to store machine code instruction bits
  - **Output port**, offset, or immediate data of some instruction will be send to data bus
  - **Flag register**, gets and sets information from / to this register
  - **Phase counter register**, is handled by computer clock and can be set to 0 by using RESET signal
  - **Two 4 bits buses**, to address each one of the output ports of the register file

An elemental computer structure (VIII)
Control Unit (and II)



Signal	Activated by	Name	Meaning
LI	Falling edge	*Load instruction*	Load machine code from data bus
DIT	Level	*Data inst. transfer*	An Offset or immediate data is transferred to data bus
LFlags	Falling edge	*Load flags*	Sets flags register after execute a logical or arithmetical instruction
RESET	Falling edge	*Reset*	Phase counter register is set to 0

An elemental computer structure (and IX)

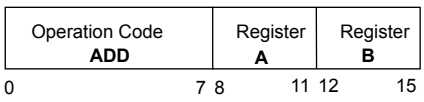


Timming

- **Computer behaviour is governed by the computer clock**
- **Instruction execution is divided in:**
  - 1. Instruction fetch
  - 2. Instruction decoding
  - 3. Instruction execution and:
    - 3.1 Operands gathering
    - 3.2 Perform operation
    - 3.3 Store the result
    - 3.4 Flag register update
- **PC update**
- **Each one of above phases takes one or more clock cycles**

Instructions execution (I)
ADD A, B (I)

- Add operation with register-register addressing mode
- Instruction format is as following:



- Instruction size: 2 bytes
- Meaning: A:= A + B;

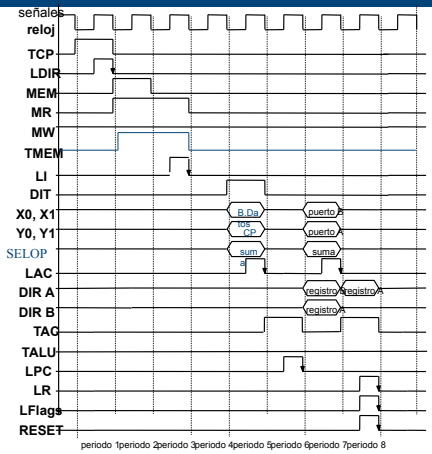
Instructions execution (II)
ADD A, B (II)

ACTION	ELEMENTAL OPERATION	CONTROL SIGNAL
Read instruction:		
▪ Addressing	CP → addresses bus	TCP
	Load Addresses Register	LDIR
▪ Reading	Start memory cycle	MEM
	Read	RD
	Open buffer to data bus	TMEM
	Load instruction machine code on Instruction Register	LI
Decoding:		Delay (1 period)
Update PC:		
▪ Add PC+2	UC put 2 on data bus	DIT
	Operands and operation selected	X0, X1 (B.Datos), Y0, Y1 (CP), Selop (suma)
	Load Accumulator Register	LAC
▪ Store new PC value	Transfer Accumulator content to data bus	TAC
	Store new PC value on PC register	LPC

Instructions execution (III)
ADD A, B (III)

ACTION	ELEMENTAL OPERATION	CONTROL SIGNAL
Performing add operation:		
▪ Get registers addresses	UC generates registers A and B addresses	DIR A (register B), DIR B (register A)
▪ Add	Operands and operation selection	X0, X1 (salida B), Y0, Y1 (salida A), Selop (suma)
	Load accumulator	LAC
Store result:		
▪ Transfer data from accumulator	Accumulator → data bus	TAC
▪ Store in A register	UC generates A register address	DIR A (register A)
	Load register file (on A register)	LR
Update flag register:		LFlags
Set to 0 to the counter phases register		RESET

Instructions execution (IV)
ADD A, B (IV)

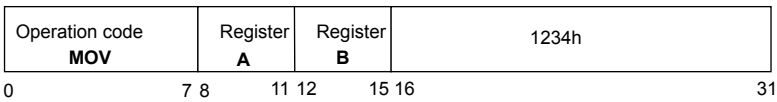


Instructions execution (V)
ADD A, B (y V)

- **Improving remarks:**
  - Fetch phase only gets one byte from memory in one byte data bus length. So, two read operations must be done.
  - 4th period is lost on decoding instruction machine code.
  - Pc is updated on 5th and 6th periods. These periods can be avoid if an adder exists only for the Addressing Unit.
  - PC update takes one period if ALU results directly goes to addresses bus or to data bus instead of storing them on Accumulator Register
  - We can overlap current instruction end phase with fetch of following instruction
  - Mind conflicts on buses

Instructions execution (VI)
MOV A, [B + 1234h] (I)

- Relative to register addressing mode
- Instruction format:



- Instruction length: 4 bytes
- Meaning: A:= Memory (B + 1234h);

Instructions execution (VII)
MOV A, [B + 1234h] (II)

ACTION	ELEMENTAL OPERATION	CONTROL SIGNAL
Fetch Address	CP → addresses bus Load addresses registere	TCP LDIR
Read instruction	Start cycle of memory Read Memory buffer → data bus Load instruction register	MEM RD TMEM LI
Decoding		1 periodo de retardo

Instructions execution (VIII)
MOV A, [B + 1234h] (III)

ACTION	ELEMENTAL OPERATION	CONTROL SIGNAL
Update PC Add CP + 4	UC puts 4 on data bus Operands and operation selection Load accumulator	DIT <X0, X1 = B. Dat> <Y0, Y1 = CP> <Selop = sumar> LAC
Store new PC value	Accumulator → Data bus Load PC register	TAC LCP

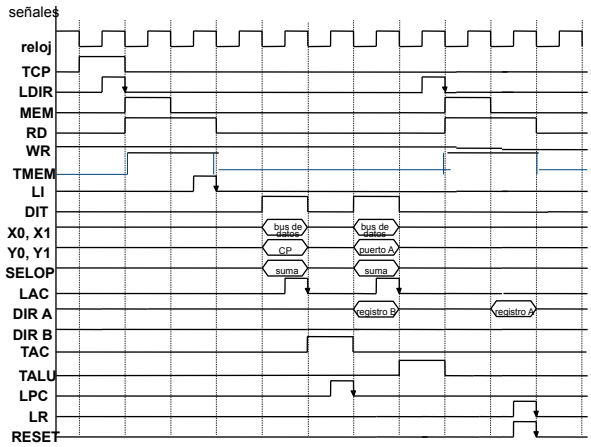
Instructions execution (IX)
MOV A, [B + 1234h] (IV)

ACTION	ELEMENTAL OPERATION	CONTROL SIGNAL
Calculate source operan address B selection + offset	UC puts 1234h. → Data bis UC generates B address	DIT <DIR A = B>
Add	Operands and operation selection Load accumulator	<X0, X1 = b.dat> <Y0,Y1 = Sal A> <Selop = suma> LAC

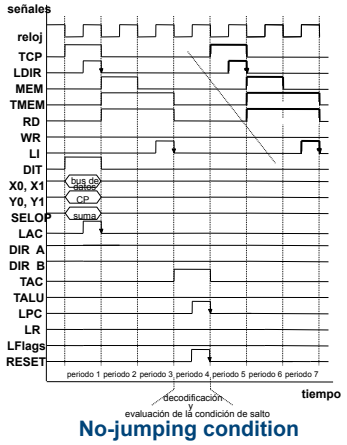
Instructions execution (X)
MOV A, [B + 1234h] (V)

ACTION	ELEMENTAL OPERATION	CONTROL SIGNAL
Store result Put source operand on data bus	Transfer accumulator to addresses bus Load addresses register Start memory cycle Read Buffer memory → Data bus UC generates A address Load register of the registers file	TALU LDIR MEM RD TMEM <DIR A = A> LR
Set to 0 phase counter		RESET

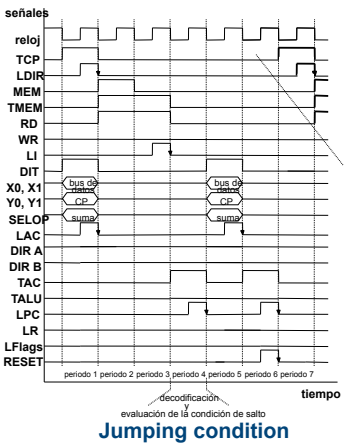
Instructions execution (XI)
MOV A, [B + 1234h] (y VI)



Instructions execution (and XII)
JZ EsCero



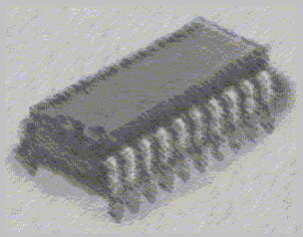
No-jumping condition



Jumping condition

Lesson 3. The Control Unit

Computers Structure and Organization



Graduated in Computer Sciences
Graduated in Computers Engineering

Academic course 2011-2012

Contents

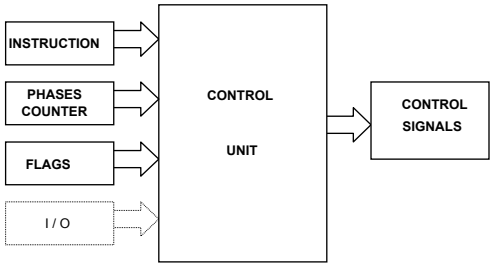
- Elemental operations
- Basic computer
  - Main Memory
  - Registers files
  - Arithmetical-Logical Unit
  - Addressing Unit
  - Control Unit
- Control signals timming
- Instructions execution
- Unit Control Design: wired vs. microprogrammed
- Microprogramming and nanoprogramming: Motorola 68000
- Computer exceptions
- Booting up the computer
- Bibliography

Contents

- Elemental operations
- Basic computer
  - Main Memory
  - Registers files
  - Arithmetical-Logical Unit
  - Addressing Unit
  - Control Unit
- Control signals timming
- Instructions execution
- Unit Control Design: wired vs. microprogrammed
- Microprogramming and nanoprogramming: Motorola 68000
- Computer exceptions
- Booting up the computer
- Bibliography

Control Unit Design (I)

- Operation code, flags and I / O signals are needed by de CU
- CU generate control signals to perform instructions executions
- CU is implemented by defining all control signal for the whole possible inputs



Control Unit Design (II)

- **Wired CU** (wired logic)
- Logic design method
- **Examples**:
- Alpha 21164, 21264, AMD K6, Pentium
- Simple logic computers
- **Advantages**:
  - Faster circuits than stored ones
- **Drawbacks**:
  - Difficult to design
  - Difficult to modify

Microprogrammed CU (stored logic)

- Control words are stored in Control Memory. Each control words enable / disable control signals of one period

Examples:

- I80x86 computers

Advantages:

  - Instruction set can be modified
  - Same computer can have several instruction set. But only one at once

Drawback:

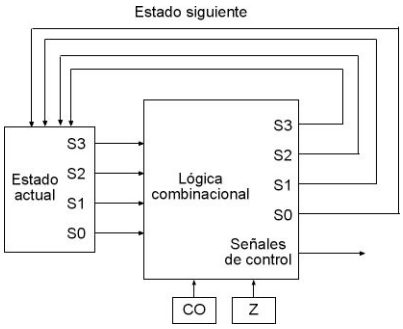
  - Slow decoding

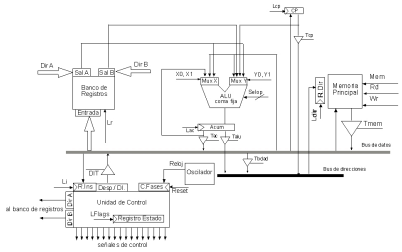
Control Unit Design (III)
Wired logic (I)

- Control Unit is a Moore machin where outputs are function of inputs and current state.
- The number of states depends on the number of elemental operations needed to perform the whole instructions set.
- CU is in an Idle State and needs a signal to start.
- CU implementation can be done:
  - **States table method**. Depending on the current state and the input signals next state is set and control signal generated
  - **Delay cells method**. In the state table diagram, states are replaced by delay cells
  - **Counter-sequencer method**. A cycles counter is used to activate / deactivate control signal of each period

Control Unit Design (IV)
Wired logic (II). States machine (I)

- Control Unit design will be:





Control Unit Design (V)
Wired logic (III). States machine (II)

- RISC instruction set is used in previous simple computer. Execution model register-register
- Wished instruction set:
  - **Memory instructions**:
    - LD R, Memory (R ← Memory content)
    - ST Memory, R (Memory Position ← R content)
  - **Process instructions**:
    - ADDI / SUBI R1, R2, immediate (R1 = R2 +/- immediate datum)
    - ADD / SUB R1, R2, R3 (R1 = R2 +/- R3)
  - **Control flow instructions:**
    - JZ label (CP ← CP + offset)
    - JMP label) CP ← label address)
- Allowed addressing modes::
  - **Immediate**, process instructions only
  - **Relative to register**, memory instructions and conditional jump only
  - **Direct to memory**, unconditional jump only

Control Unit Design (VI)
Wired logic (IV). States machine (III)

- Instruction format 32 bits.
- Operation codes are:

Instruction	Operation Code		
	O3	O2	O1
LD	0	0	0
ST	0	0	1
ADD R1, R2, R3	0	1	0
SUB R1, R2, R3	0	1	1
ADDI R1, R2, Imm	1	0	0
SUBI R1, R2, Imm	1	0	1
JZ Label	1	1	0
JMP Label	1	1	1

Memory instructions

CO	R	Offset
(3)	(4)	(25)

Process instructions

CO	R1	R2	Immediate
(3)	(4)	(4)	(21)

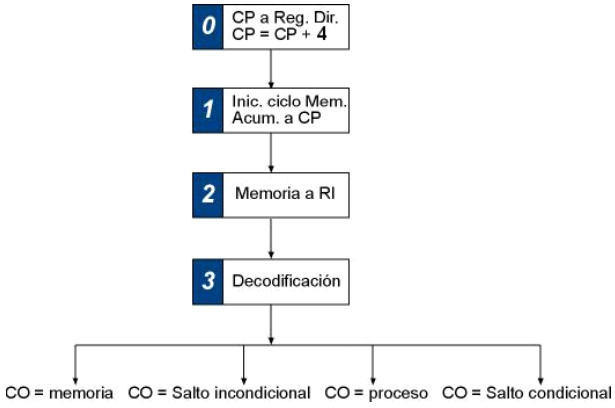
CO	R1	R2	R3	Not Used
(3)	(4)	(4)	(4)	(17)

Conditional / Unconditional Jump Instructions

CO	Offset / Direct memory address
(3)	(29)

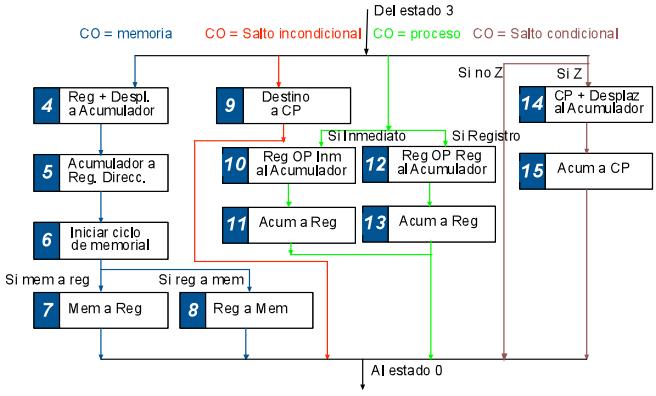
Control Unit Design (VII)
Wired logic (V). States machine (IV)

- States machine (first part)



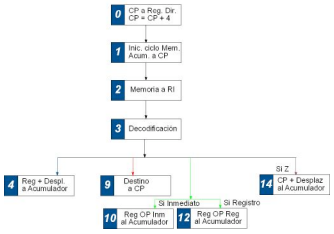
Control Unit Design (VIII)
Wired logic (VI). States machine (V)

- States machine (second part)



Control Unit Design (IX)
Wired logic (VII). States machine (VI)

- Transition between states function must be defined
- 4 bits are needed to number 15 states
- State #0 (0000), state #1 (0001), state #2 (0010), etc

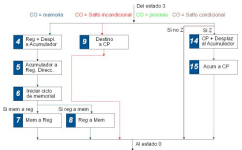


- Transition states equations:

New State					Condition	Ecuation
State	S3	S2	S1	S0		
1	0	0	0	1	Previous State 0	$S0 = \overline{S3} \cdot \overline{S2} \cdot \overline{S1} \cdot S0$
2	0	0	1	0	Previous State 1	$S1 = \overline{S3} \cdot \overline{S2} \cdot S1 \cdot \overline{S0}$
3	0	0	1	1	Previous State 2	$S1 = \overline{S3} \cdot \overline{S2} \cdot S1 \cdot S0$ y $S0 = \overline{S3} \cdot \overline{S2} \cdot S1 \cdot \overline{S0}$
4	0	1	0	0	Previous State 3 And CO=memory	$S2 = \overline{S3} \cdot \overline{S2} \cdot S1 \cdot \overline{S0} (\overline{O2} \cdot O1 \cdot \overline{O0} + \overline{O2} \cdot \overline{01} \cdot O0)$

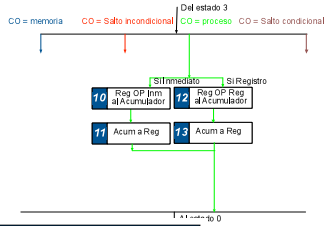
Control Unit Design (X)
Wired logic (VIII). States machine (VII)

- Transition states equations (cont):



Estado nuevo					Condición	Ecuación
Estado	S3	S2	S1	S0		
5	0	1	0	1	Estado Anterior 4	$S2 = \overline{S3} \cdot S2 \cdot \overline{S1} \cdot \overline{S0}$ y $S0 = \overline{S3} \cdot S2 \cdot \overline{S1} \cdot \overline{S0}$
6	0	1	1	0	Estado Anterior 5	$S2 = \overline{S3} \cdot S2 \cdot \overline{S1} \cdot S0$ y $S1 = \overline{S3} \cdot S2 \cdot \overline{S1} \cdot S0$
7	0	1	1	1	Estado Anterior 6 Y CO = ST	$S2 = S1 = S0 = \overline{S3} \cdot S2 \cdot S1 \cdot \overline{S0} \cdot \overline{O2} \cdot \overline{O1} \cdot O0$
8	1	0	0	0	Estado Anterior 6 Y CO = LD	$S3 = \overline{S3} \cdot S2 \cdot S1 \cdot \overline{S0} \cdot \overline{O2} \cdot \overline{O1} \cdot \overline{O0}$
9	1	0	0	1	Estado Anterior 3 Y CO = Salto incondicional	$S3 = S0 = \overline{S3} \cdot \overline{S2} \cdot S1 \cdot \overline{S0} \cdot O2 \cdot \overline{O1} \cdot \overline{O0}$
14	1	1	1	0	Estado Anterior 3 Y CO = Salto condicional Y Z	$S3 = S2 = S1 = \overline{S3} \cdot \overline{S2} \cdot S1 \cdot \overline{S0} \cdot O2 \cdot \overline{O1} \cdot \overline{O0} \cdot Z$
15	1	1	1	1	Estado Anterior 14	$S3 = S2 = S1 = S0 = \overline{S3} \cdot \overline{S2} \cdot \overline{S1} \cdot \overline{S0}$

Control Unit Design (XI)
Wired logic (IX). States machine (VIII)



- Transition states equations:

New State					Condition	Ecuation
State	S3	S2	S1	S0		
10	1	0	1	0	Previous State 3 And CO = ADDI or SUBI	$S3 = S1 = \overline{S3}\cdot\overline{S2}\cdot S1\cdot S0\cdot(\overline{O2}\cdot O1\cdot O0 + \overline{O2}\cdot O1\cdot\overline{O0})$
11	1	0	1	1	Previous State 10	$S3 = S1 = S0 = S3\cdot\overline{S2}\cdot S1\cdot S0$
12	1	1	0	0	Previous State 3 And CO = ADD or SUB	$S3 = S2 = \overline{S3}\cdot\overline{S2}\cdot S1\cdot S0\cdot(\overline{O2}\cdot O1\cdot\overline{O0} + \overline{O2}\cdot O1\cdot O0)$
13	1	1	0	1	Previous State 12	$S3 = S2 = S0 = S3\cdot S2\cdot\overline{S1}\cdot S0$
0	0	0	0	0	Previous State 7or 8 or 9 or 11 or 13 or15 or (3 and not Z)	$S3 = S2 = S1 = S0 = \overline{S3}\overline{S2}S1S0 + S3\overline{S2}S1S0 + S3S2\overline{S1}S0 + + S3\overline{S2}S1S0 + S3S2\overline{S1}S0 + S3\overline{S2}S1S0 + S3S2S1S0\overline{Z}$

Control Unit Design (XII)
Wired logic (X). States machine (IX)

- Generated signals:

State	Elemental Op.	Signals
0	Address Reg. ←PC	TCP, LDIR
	PC = PC + 4	MUX Y, DIT, MUX X, SELOP, LAC
1	Start Memory Cycle	MEM, RD
	PC ← Accumulator	TAC, LCP
2	RI ← Memory	RD, TMEM, LI
3	Decoding	Anything but checking flags
4	Accumulator ← Register + Offset	DIR A, DIT, MUX X, MUX Y, SELOP, LAC
5	Address Reg. ←Accum	TALU, LDIR
6	Start Memory Cycle	MEM
7	Register ← Memory	RD, TMEM, DIRA, LR, RESET
8	Memory ← Register	DIR A, DIR B, MUX X, MUX Y, SELOP, TAC, WR, RESET

Control Unit Design (XIII)
Wired logic (XI). States machine (X)

- Generated signals:

State	Elemental Op.	Signals
9	PC ← Target	DIT, LCP, RESET
10	Accumulator ← Register OP Immediate	DIR A, DIT, MUX Y, MUX X, SELOP, LAC, LFLAGS
11	Register ← Accumulator	TAC, DIR A, LR, RESET
12	Accumulator ← Register OP Register	DIR A, DIR B, MUX Y, MUX X, SELOP, LAC, LFLAGS
13	Register ← Accumulator	TAC, DIR A, LR, RESET
14	Accumulator ← PC + offset	MUX Y, DIT, MUX X, SELOP, LAC
15	PC ← Accumulator	TAC, LCP, RESET

Control Unit Design (XIV)
Wired logic (XII). States machine (XI)

- CU moves throught states machine when transition states equations are defined.
- Several different states generate same output signals
- LCP signal is activated in 1, 9 and 15 states. LAC signal is activated in 0, 4, 10, 12 and 14 states. Signals equations are:

Signal	States in which signal is activated	Signal output equation
LCP	1, 9 and 15	$LCP = \overline{S3}\cdot\overline{S2}\cdot\overline{S1}\cdot S0 + S3\cdot\overline{S2}\cdot\overline{S1}\cdot S0 + S3\cdot S2\cdot\overline{S1}\cdot S0$
LAC	0, 4, 10, 12 and 14	$LAC = \overline{S3}\cdot\overline{S2}\cdot\overline{S1}\cdot\overline{S0} + \overline{S3}\cdot\overline{S2}\cdot\overline{S1}\cdot S0 + S3\cdot\overline{S2}\cdot\overline{S1}\cdot S0 + S3\cdot\overline{S2}\cdot\overline{S1}\cdot\overline{S0} + S3\cdot S2\cdot\overline{S1}\cdot\overline{S0}$
….	…..	…..

Control Unit Design (XV)
Wired logic (XIII). States machine (XII)

- Signals equations:

Signal	States in which signal is activated	Signal output equation
LCP	1, 9 and 15	$LCP = \overline{S3}\cdot S2\cdot\overline{S1}\cdot\overline{S0} + S3\cdot S2\cdot\overline{S1}\cdot\overline{S0} + S3\cdot\overline{S2}\cdot S1\cdot\overline{S0}$
TCP	0	$TCP = \overline{S3}\cdot\overline{S2}\cdot\overline{S1}\cdot\overline{S0}$
DIR A	4, 7, 8, 10, 11, 12 and 13	$DIRA = \overline{S3}S2\overline{S1}\overline{S0} + \overline{S3}S2S1\overline{S0} + S3\overline{S2}\overline{S1}\overline{S0} + S3\overline{S2}S1\overline{S0} + S3S2\overline{S1}\overline{S0} + S3S2S1\overline{S0}$
DIR B	8 and 12	$DIRB = S3\cdot\overline{S2}\cdot\overline{S1}\cdot\overline{S0} + S3\cdot S2\cdot\overline{S1}\cdot\overline{S0}$
LR	7, 11 and 13	$LR = \overline{S3}\cdot S2\cdot S1\cdot S0 + S3\cdot\overline{S2}\cdot S1\cdot S0 + S3\cdot S2\cdot\overline{S1}\cdot S0$
MUX X	0, 4, 8, 10, 12 and 14	$MUX X = \overline{S3}\cdot\overline{S2}\cdot\overline{S1}\cdot\overline{S0} + \overline{S3}\cdot S2\cdot\overline{S1}\cdot\overline{S0} + S3\cdot\overline{S2}\cdot\overline{S1}\cdot\overline{S0} + S3\cdot S2\cdot\overline{S1}\cdot\overline{S0} + S3\cdot\overline{S2}\cdot S1\cdot\overline{S0} + S3\cdot S2\cdot S1\cdot\overline{S0}$
MUX AND	0, 4, 8, 10, 12 and 14	$MUX Y = \overline{S3}\cdot\overline{S2}\cdot\overline{S1}\cdot\overline{S0} + \overline{S3}\cdot S2\cdot\overline{S1}\cdot\overline{S0} + S3\cdot\overline{S2}\cdot\overline{S1}\cdot\overline{S0} + S3\cdot S2\cdot\overline{S1}\cdot\overline{S0} + S3\cdot\overline{S2}\cdot S1\cdot\overline{S0} + S3\cdot S2\cdot S1\cdot\overline{S0}$
SELOP	0, 4, 8, 10, 12 and 14	$SELOP = \overline{S3}\cdot\overline{S2}\cdot\overline{S1}\cdot\overline{S0} + \overline{S3}\cdot S2\cdot\overline{S1}\cdot\overline{S0} + S3\cdot\overline{S2}\cdot\overline{S1}\cdot\overline{S0} + S3\cdot S2\cdot\overline{S1}\cdot\overline{S0} + S3\cdot\overline{S2}\cdot S1\cdot\overline{S0} + S3\cdot S2\cdot S1\cdot\overline{S0}$
LAC	0, 4, 10, 12 and 14	$LAC = \overline{S3}\cdot\overline{S2}\cdot\overline{S1}\cdot\overline{S0} + \overline{S3}\cdot S2\cdot\overline{S1}\cdot\overline{S0} + S3\cdot\overline{S2}\cdot\overline{S1}\cdot\overline{S0} + S3\cdot S2\cdot\overline{S1}\cdot\overline{S0} + S3\cdot\overline{S2}\cdot S1\cdot\overline{S0}$
TAC	1, 8, 11, 13 and 15	$TAC = \overline{S3}\cdot\overline{S2}\cdot S1\cdot\overline{S0} + \overline{S3}\cdot S2\cdot S1\cdot\overline{S0} + S3\cdot\overline{S2}\cdot S1\cdot\overline{S0} + S3\cdot S2\cdot S1\cdot\overline{S0} + S3\cdot\overline{S2}\cdot S1\cdot\overline{S0}$
TALU	5	$TALU = \overline{S3}\cdot\overline{S2}\cdot\overline{S1}\cdot\overline{S0}$

Control Unit Design (XVI)
Wired logic (XIV). States machine (and XIII)

- Signals equations:

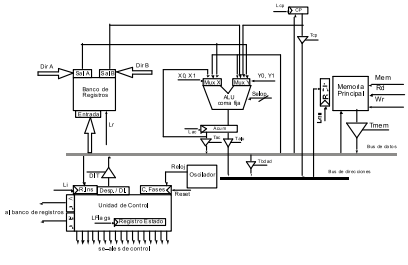
Signal	States in which signal is activated	Signal output equation
DIT	0, 4, 9, 10 and 14	$DIT = \overline{S3}\cdot\overline{S2}\cdot\overline{S1}\cdot\overline{S0} + \overline{S3}\cdot S2\cdot\overline{S1}\cdot\overline{S0} + S3\cdot\overline{S2}\cdot\overline{S1}\cdot\overline{S0} + S3\cdot\overline{S2}\cdot S1\cdot\overline{S0} + S3\cdot S2\cdot\overline{S1}\cdot\overline{S0}$
LI	2	$LI = \overline{S3}\cdot S2\cdot\overline{S1}\cdot\overline{S0}$
RESET	7, 8, 9, 11, 13 and 15	$RESET = \overline{S3}S2S1S0 + S3\overline{S2}S1S0 + S3S2\overline{S1}S0 + S3S2S1\overline{S0} + S3S2\overline{S1}S0 + S3S2S1S0$
LFLAGS	10 and 12	$LFLAGS = S3\cdot\overline{S2}\cdot S1\cdot\overline{S0} + S3\cdot S2\cdot\overline{S1}\cdot\overline{S0}$
MEM	1 and 6	$MEM = \overline{S3}\cdot\overline{S2}\cdot\overline{S1}\cdot S0 + \overline{S3}\cdot S2\cdot\overline{S1}\cdot\overline{S0}$
RD	2 and 7	$RD = \overline{S3}\cdot\overline{S2}\cdot S1\cdot\overline{S0} + \overline{S3}\cdot S2\cdot S1\cdot\overline{S0}$
WR	8	$WR = S3\cdot\overline{S2}\cdot\overline{S1}\cdot\overline{S0}$
LDIR	0 and 5	$LDIR = \overline{S3}\cdot\overline{S2}\cdot\overline{S1}\cdot\overline{S0} + \overline{S3}\cdot S2\cdot\overline{S1}\cdot S0$
TMEM	2 and 7	$TMEM = \overline{S3}\cdot\overline{S2}\cdot S1\cdot\overline{S0} + \overline{S3}\cdot S2\cdot S1\cdot\overline{S0}$

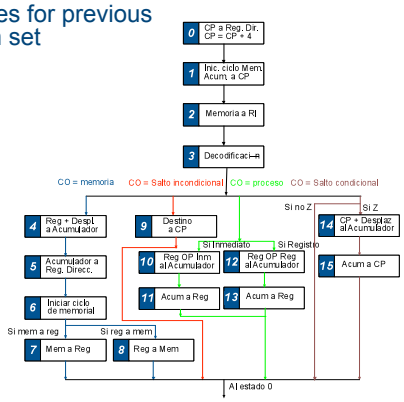
Control Unit Design (XVII)
Wired logic (XV). Delay cells (I)

- States machine diagram is implemented by using daly cells.
- Control signals are attached to delay cells
- Delay cells sinchronized control signals generation
- One only pulse is introduced in the system
- Pulse circulates thrught the circuit and activates control signals when required.

Control Unit Design (XVIII)
Wired logic (XVI). Delay cells (II)

- LR is activated in 7, 11 and 13 states for previous elemental computer and instruction set

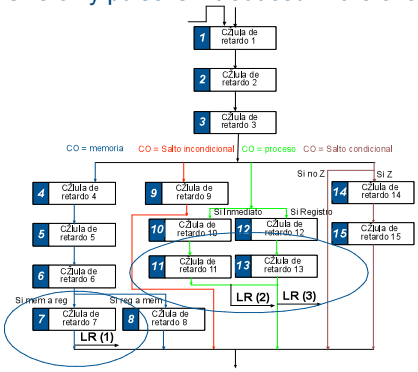


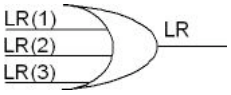


Signal	States in which signal is activated	Signal output equation
LR	7, 11 and 13	$LR = S3 \cdot \overline{S2} \cdot \overline{S1} \cdot S0 + S3 \cdot \overline{S2} \cdot S1 \cdot S0 + S3 \cdot S2 \cdot \overline{S1} \cdot \overline{S0}$

Control Unit Design (XIX)
Wired logic (XVII). Delay cells (and III)

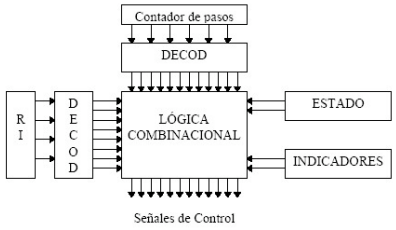
- One only pulse is introduced in the circuit





Control Unit Design (XX)
Wired logic (XVIII). Sequencier

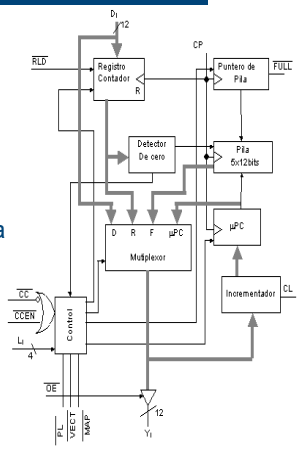
- A counter is used to establish when signals must be activated
- Control unit must also know:
  - Current instruction
  - Flags information
- Using above information, instruction will be executed by activating control signal when needed.



Control Unit Design (XXI)
Wired logic (XIX). Sequiencier (II)

AMD 2910. Specifications

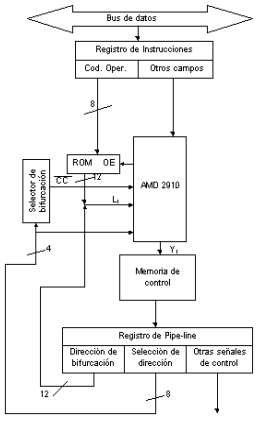
- Microinstruction counter with self increment
- Five level micro stack
- Counter with zero detection
- Next instruction address is set via multiplexor inputs
- Conditional micro jumps are possible via condition signals input



Control Unit Design (XXII)
Wired logic (and XX) Sequiencier (and III)

IBM 370 / 45 Control Unit implementation by using an AMD2910

- AMD 2910 based microprocessor
- 32 bits length instruction format
- 8 bits operation code



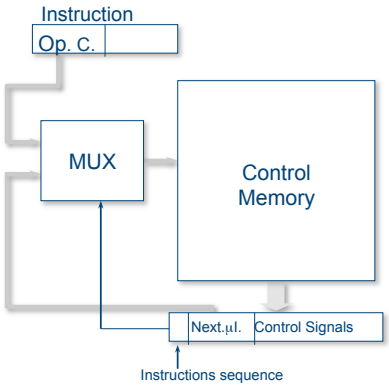
Control Unit Design (XXIII) Microprogrammed logic (I)

- A set of microinstructions is a microprogram to execute an instruction
- Firmware is the set of all microprogram to execute the whole instructions of an instruction set.
- To be taken into account:
  - Control memory size
  - Each instruction must be related to its microprogram
  - Microinstruction control sequence

Control Unit Design (XXIV) Microprogrammed logic (II)

Explicit sequence of instructions:

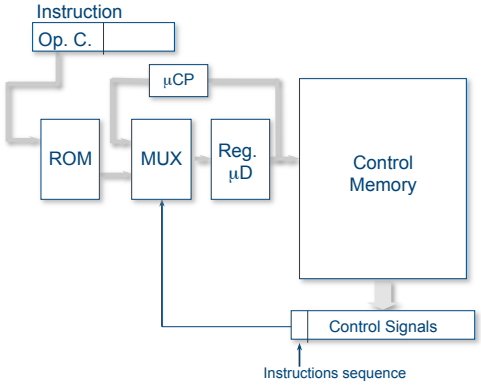
- Each µI. has a field to store next µI address.
- Operation code points to starting micro program memory address.
- Each µI has an ending bit microprogram.
- **Advantage**: different microprograms can use common µIs. sequences
- **Drawback**: Required Control Memory Word Size to store next µI addresses



Control Unit Design (XXV) Microprogrammed logic (III)

Implicit sequence of instructions:

- The whole µIs. belonging to a microprogram are sequential ordered.
- It's needed:
  - A microprogram counter (µCP) to point to subsequence µIs.
  - First µI. address of the microprogram must be stored on a ROM-



Control Unit Design Control (XXVI) Microprogrammed logic (IV)

µinstructions decoding

- **Horizontal µprogramming,** when codification is not used
- **Vertical µprogramming**, µIs. Are codified

Horizontal µprogramming

- µIs. Perform an elemental operation per cycle only
- Each one of the *n* bits straight controls a hardware element
- µIs. have long formats
- µIs. are fast
- The main drawbacks is that memory is disperse.

Vertical µprogramming

- Control memory capacity decrease
- µIs. are slow because decoding is neede

Control Unit Design (XXVII) Microprogrammed logic (V)

µinstrucciones coding

Control signals are grouped in:

- Data bus access
- Addresses bus access
- ALU control signals
- Registers file control signals
- Memory control signals
- Addressing Unit control signals
- Flags
- I / O control signals

Data Bus	Addresses Bus	ALU	Registers File	Memory	Addressing Unit
TMEM	TALU	SELOP	LR	MEM	LPC
TAC	TCP	X0, X1	DIR. A	RD	
DIT	TBDAD	Y0, Y1	DIR. B	WR	
		LAC		LDIR	

Control Unit Design (XXVIII) Microprogrammed logic (VI)

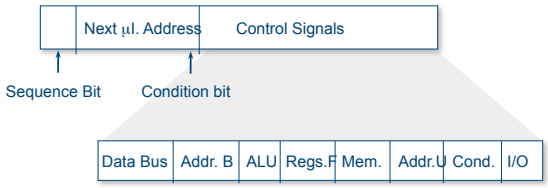
Conditional micro jumps

- Depending on condition, conditional instructions have two possible chronograms.
- A micro jump needs a mechanism to decide where to jump.
- Depending on instruction sequence different mechanisms will be applied.

Control Unit Design (XXIX)
Microprogrammed logic (VII)

Conditional micro jumps. Explicit sequence of instructions:

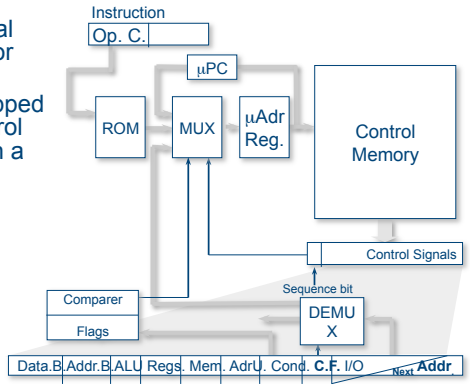
- Each micro instruction has next micro instruction address
- Instruction format will be increased a lot if two addresses required. Solution is differentiate both addresses in one bit only
- Condition bit will be 1 if condition is true or 0 if false



Control Unit Design (XXX)
Microprogrammed logic (VIII)

Conditional micro jumps. Implicit sequence of instructions:

- To choose between normal sequence of instructions or another µI,
- Next µI address is overlapped with another / others control signals field depending on a control bit
- C.F.: Control bit



Control Unit Design (XXXI)
Microprogrammed logic (IX)

- A control memory has a lot of micro instructions.
- Nanoprogramming allows to remove duplicate microinstruction by implementing control memory in a two levels control memory. All microinstruction of the second level memory will be different.
- Second level control words are called nanoinstructions.
- Can be grouped in micro procedures
- It's recommended to use nanoprograming if:

$$n \cdot m < (n \cdot \log_2 \cdot d) + (d \cdot m)$$

Where :

n is the number of microinstructions in the control memory

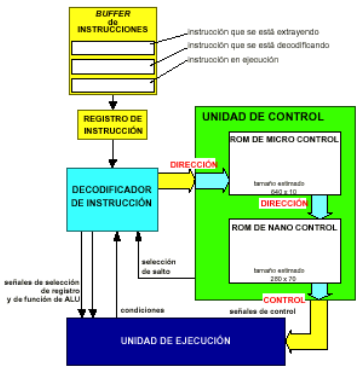
m is the length of each microinstruction (in bits)

d is the number of different microinstructions

Control Unit Design (XXXII)
Microprogrammed logic (and X)

M68000 nanoprogramming

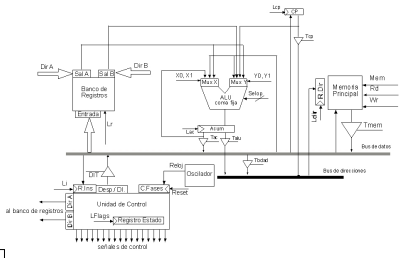
- Two-level nanoprogramming system:
  - First level words are pointers to second level ones
  - Second level words are differents
  - Second level words activate / desactivate a hardware piece (horizontal microprogrammig)
- Second level control word is 70 bits length



Control Unit Design (XXXIII)
Microprogrammed Computer Design (I)

Registers load signals

- Reset
- Lflags
- Li
- Lr
- Lac
- Ldir

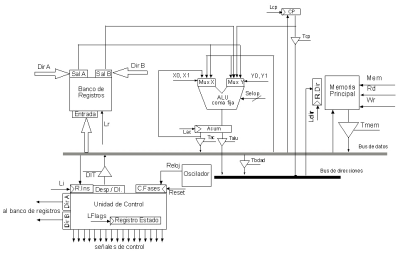


Señales de carga de registro					
5	4	3	2	1	0
Ldir	Lac	Lr	Li	Lflags	Reset

Control Unit Design (XXXIV)
Microprogrammed Computer Design (II)

Data bus access control signals

- Tmem
- Tac
- Dit



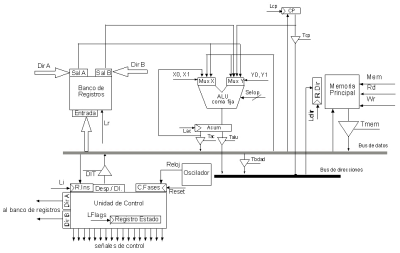
B.Datos	
7	6

B7	B6	
0	0	Tmem
0	1	Tac
1	0	Dit
1	1	Nop

Control Unit Design (XXXV)
Microprogrammed Computer Design (III)

Addresses bus access control signals

- Tcp
- Talu
- Tbdad



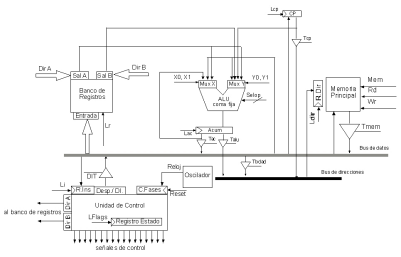
B.Direcciones	
9	8

B9	B8	
0	0	Tcp
0	1	Talu
1	0	Tbdad
1	1	Nop

Control Unit Design (XXXVI)
Microprogrammed Computer Design (IV)

ALU control signals

- X1, X0
- Y1, Y0
- Selop



Selop				Y1	Y0	X1	X0
17	16	15	14	13	12	11	10

B11	B10	
0	0	Acumulador
0	1	B. datos
1	0	Salida A
1	1	Nop

B13	B12	
0	0	CP
0	1	B. datos
1	0	Salida A
1	1	Salida B

B17	B16	B15	B14	
0	0	0	0	ADD
0	0	0	1	SUB
0	0	1	0	MUL
0	0	1	1	DIV
...	...	...	...	...
1	1	1	1	XOR

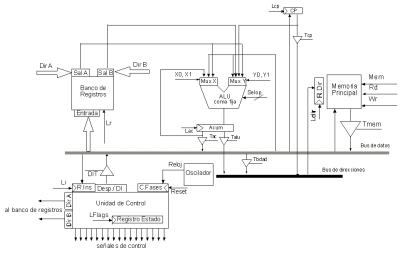
Control Unit Design (XXXVII)
Microprogrammed Computer Design (V)

Memory control signals

- Mem
- Rd
- Wr

Mem	Rd/Wr
19	18

B19	B18	
0	0	Nop
0	1	Nop
1	0	Escribir
1	1	Leer



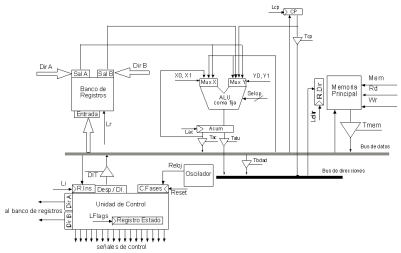
Control Unit Design (XXXVIII)
Microprogrammed Computer Design (VI)

Addressing Unit control signals

- Lcp

Unidad Direccionamiento
20

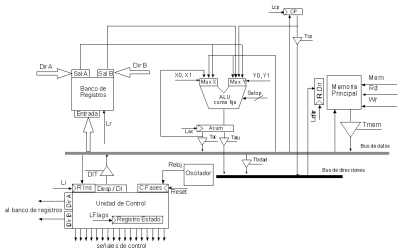
B20	
0	Nop
1	Cargar CP



Control Unit Design (XL)
Microprogrammed Computer Design (VIII)

Remaining bits:

- Condition bit
- Micro jump bit
- End micro program bit / sequence of instruction bit



Fin microprograma	Microsalto	Condición
31	30	29

B29	
0	No cumple la condición
1	Si cumple la condición

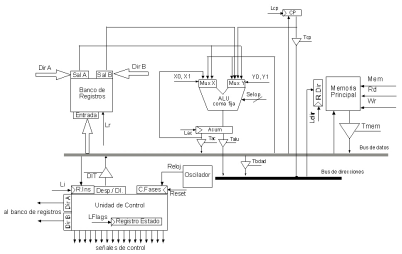
B30	
0	No lleva microsalto
1	Si lleva microsalto

B31	
0	No termina microprg
1	Fin de microprograma

Control Unit Design (XLI)
Microprogrammed Computer Design (IX)

Remaining bits:

- Lets a control memory of 1K. It will be necessary 10 bits to address it.
- Depending on 29th and 30th bits value, bits numbered from 0 to 9 will be a micro jump address or registers load and data bus access control signals by using fields overlapping.



Control Unit Design (and XLII)
Microprogrammed Computer Design (and X)

Micro instruction without micro jump

Fin microprograma	Microsalto	Condición	Banco de registros			Unidad. Direccionamiento	Memoria		ALU			Bus de direcciones		Bus de datos		Carga de registros		
31	30	29	28	...	21	20	19	18	17	...	10	9	8	7	6	5	...	0

Micro instruction with micro jump

Fin microprograma	Microsalto	Condición	Banco de registros			Unidad. Direccionamiento	Memoria		ALU			Dirección de microsalto dependiendo del valor que tengan B30 y B29						
31	30	29	28	...	21	20	19	18	17	...	10	9	.....					0

Exceptions

- Exceptions are unexpected events in microprocessor.
- Exceptions are asynchronous signals (divide by zero, parity error, overflow, operating system traps, etc)
- Control Unit must detect exception and be handled by the operating system as steps bellow:
  - Stop program execution
  - Send exception type to the operating system
  - Give control to the operating system
- To send exception type to the operating system we can use:
  - **Exception register**. Control Unit set exception type in a special register
  - **Vectored exceptions**. Different addresses are assigned to each exception type. Operating system executes the found procedure in such address.

Booting up the computer

- A random value is set to computer elements when starting up.
- A RESET is forced to all computer registers. PC is load with the loader program ROM address.
- This program will do:
  - Testing routines.
  - Setting up routines. (devices, assigned ports, interrupts array, etc)
  - Load vector array
  - Load operating system
  - Load device drivers

Bibliography

- Estructura y diseño de computadores
  David A. Patterson y John L. Hennessy. Reverté, 2000
  Capítulo 5 y Apéndice C
- Estructura y diseño de computadores
  Jose Mª Angulo. Paraninfo, 1996
  Capítulo 5
- Arquitectura de computadores. Un enfoque cuantitativo
  John L. Hennessy y David A. Patterson. Mc Graw Hill, 3ª ed, 2002
  Capítulo 5
- Arquitectura de computadores
  José A. de Frutos y Rafael Rico. Servicio de Publicaciones de la Universidad de Alcalá, 1995
  Capítulo 4.