COMPUTER ARCHITECTURE AND SOFTWARE EXECUTION PROCESS

MICROPROCESSOR ARCHITECTURE

Bachelor in Artificial Intelligence, Data and Management Sciences
CentraleSupelec and ESSEC Business School - 2023/2024

Idir AIT SADOUNE
idir.aitsadoune@centralesupelec.fr

OUTLINE

Back to the begin - Back to the outline

CentraleSupélec | université PARIS-SACLAY

OUTLINE

Back to the begin – Back to the outline

CentraleSupélec | université PARIS-SACLAY

REMINDER

REMINDER

- What can a computer do ?
  - copy values between storage units
  - perform logical/arithmetic operations between stored values
  - move within the program (jump), possibly conditionally

REMINDER

- What can a computer do ?
  - copy values between storage units
  - perform logical/arithmetic operations between stored values
  - move within the program (jump), possibly conditionally
- A computer executes a very low level language (**Machine Language**)
  - means of controlling and elementary control of computer electronics
  - the instructions of this language carry out elementary operations

DESCRIPTION OF AN INSTRUCTION

Some questions

DESCRIPTION OF AN INSTRUCTION

Some questions

- What **operation** to perform ?

DESCRIPTION OF AN INSTRUCTION

Some questions

- What **operation** to perform ?
- Where are the **operands** ?

DESCRIPTION OF AN INSTRUCTION

Some questions

- What **operation** to perform ?
- Where are the **operands** ?
- Where to put the **result** ?

DESCRIPTION OF AN INSTRUCTION

Some questions

- What **operation** to perform ?
- Where are the **operands** ?
- Where to put the **result** ?
- Where is **the following instruction** ?

DESCRIPTION OF AN INSTRUCTION

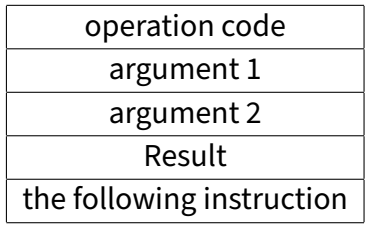
Some questions

- What **operation** to perform ?
- Where are the **operands** ?
- Where to put the **result** ?
- Where is **the following instruction** ?
- What size for memory words, instructions, and data ?

THEORETICAL INSTRUCTION

THEORETICAL INSTRUCTION

- 4-address machine



THEORETICAL INSTRUCTION

- 4-address machine

operation code
argument 1
argument 2
Result
the following instruction

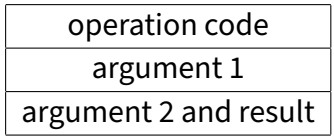
- 3-address machine (with the **PC-Program Counter** register)

operation code
argument 1
argument 2
Result

CentraleSupélec université PARIS-SACLAY

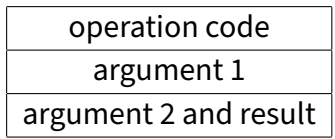
THEORETICAL INSTRUCTION

- 2-address machine (duality of an operand)

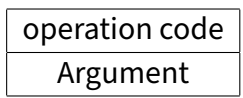


THEORETICAL INSTRUCTION

- 2-address machine (duality of an operand)



- 1-address machine (adding the **AC-Accumulator** register)

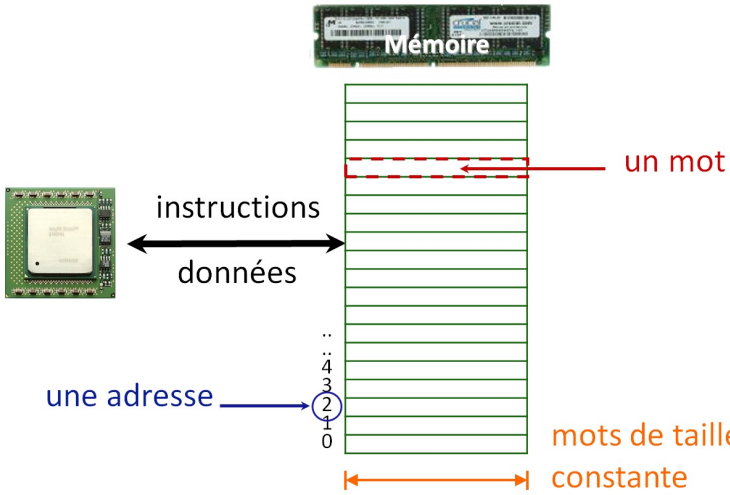


OUTLINE

Back to the begin – Back to the outline

CentraleSupélec | université PARIS-SACLAY

THE VON NEUMANN MACHINE



THE STUDIED MACHINE

The architecture

- 1-address machine
- Memory containing 16-bit words (2 bytes) : data and instructions
- Addresses also on 16 bits (2^{16} = 64 kilo words = 128 kilo bytes)

THE STUDIED MACHINE

The architecture

- 1-address machine
- Memory containing 16-bit words (2 bytes) : data and instructions
- Addresses also on 16 bits (2^{16} = 64 kilo words = 128 kilo bytes)

Operations

- loading (**LDA**) and storing (**STA**) the accumulator in memory
- addition (**ADD**) and subtraction(**SUB**)
- clearing the accumulator (**CLR**)
- jump to an instruction (**JMP** and **JAN**)

CentraleSupélec | université PARIS-SACLAY

THE STUDIED MACHINE

THE STUDIED MACHINE

Addressing modes

- **immediate** : argument = the operand (`ADD #10`)
- **direct** : argument = address of the operand (`LDA $2000`)
- **indirect** : argument = the address of the address of the operand (`STA @$3000`)
- **implicit** : no argument (`CLR`)

CentraleSupélec | université PARIS-SACLAY

THE STUDIED MACHINE

Addressing modes

- **immediate** : argument = the operand (`ADD #10`)
- **direct** : argument = address of the operand (`LDA $2000`)
- **indirect** : argument = the address of the address of the operand (`STA @$3000`)
- **implicit** : no argument (`CLR`)

Case of Jump

- **direct** jump (`JMP $2000`)
- **indirect** jump (`JMP @$3000`)

CentraleSupélec | université PARIS-SACLAY

INSTRUCTION CODING

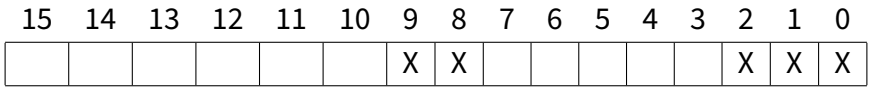
- Writing a value to an address : `Mem[$200] := 45`
- Reading the value contained at an address : `R := Mem[$20F]`

INSTRUCTION CODING

- Writing a value to an address : `Mem[$200] := 45`
- Reading the value contained at an address : `R := Mem[$20F]`

	immediate	**Direct**	**Indirect**
LDA	AC := arg	AC := Mem[arg]	AC := Mem[Mem[arg]]
STA		Mem[arg] := AC	Mem[Mem[arg]] := AC
ADD	AC := AC + arg	AC := AC + Mem[arg]	AC := AC + Mem[Mem[arg]]
SUB	AC := AC - arg	AC := AC - Mem[arg]	AC := AC - Mem[Mem[arg]]
JMP/JAN		PC := arg	PC := Mem[arg]

THE STUDIED MACHINE



00 : implicit

01 : immediate

10 : direct

11 : indirect

000 : LDA

001 : ADD

010 : SUB

011 : STA

100 : JMP

101 : JAN

110 : CLR

111 : —

AN EXAMPLE

Direct	0000	0010	0000	0000	LDA
	0010	0000	0000	0000	$2000
immediate	0000	0001	0000	0001	ADD
	0000	0000	0000	1010	$000A
Indirect	0000	0011	0000	0011	STA
	0010	0000	0000	0010	$2002

```
1 LDA $2000
2 ADD #$000A
3 STA @$2002
```

ANOTHER EXAMPLE

```
1 NB1:    WORD  #100
2 NB2:    WORD  #0
3 NB2ADR: WORD  NB2
4
5 BEGIN:  LDA NB1
6         ADD #10
7         STA @NB2ADR
```

CentraleSupélec | université PARIS-SACLAY

OUTLINE

Back to the begin - Back to the outline

CentraleSupélec | université PARIS-SACLAY

THE PROCESSOR CYCLE

THE PROCESSOR CYCLE

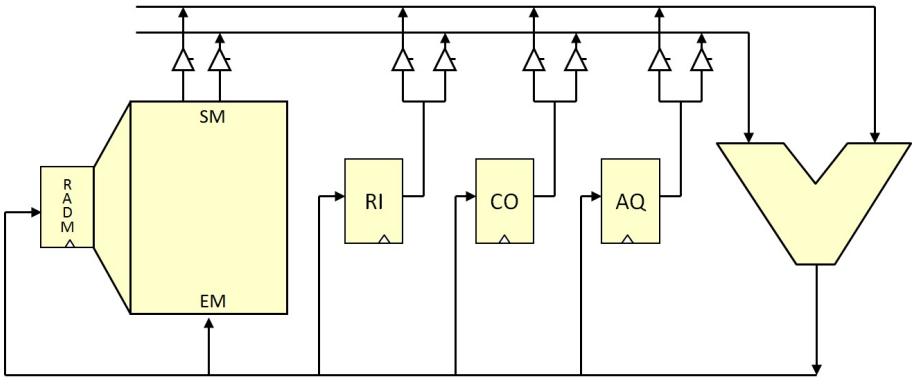
- The processor cycle takes place in 3 stages
  1. fetch the instruction
  2. fetch the operand
  3. execute the instruction

THE PROCESSOR CYCLE

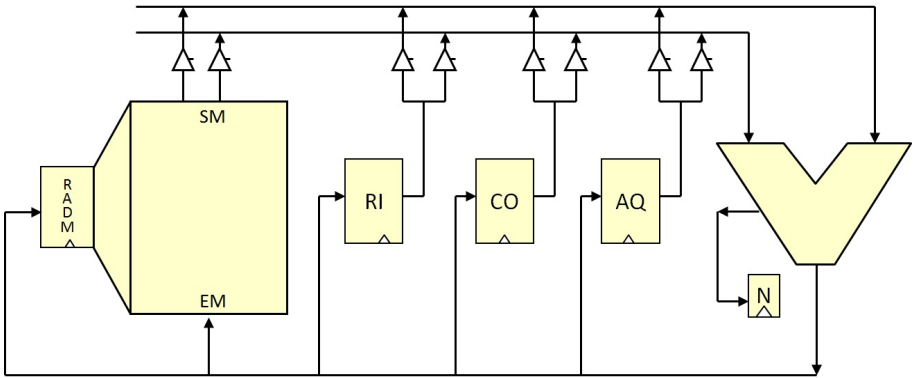
- The processor cycle takes place in 3 stages
  1. fetch the instruction
  2. fetch the operand
  3. execute the instruction

- Need to store instruction in step 1 to know what to do in step 3
  - instruction register (**IR**)

A GENERIC DATA PATH



ADAPTATION TO OUR PROCESSOR



OUTLINE

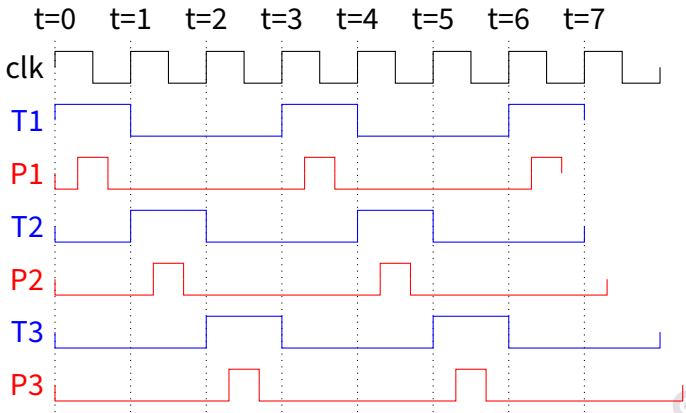
Back to the begin - Back to the outline

CentraleSupélec | université PARIS-SACLAY

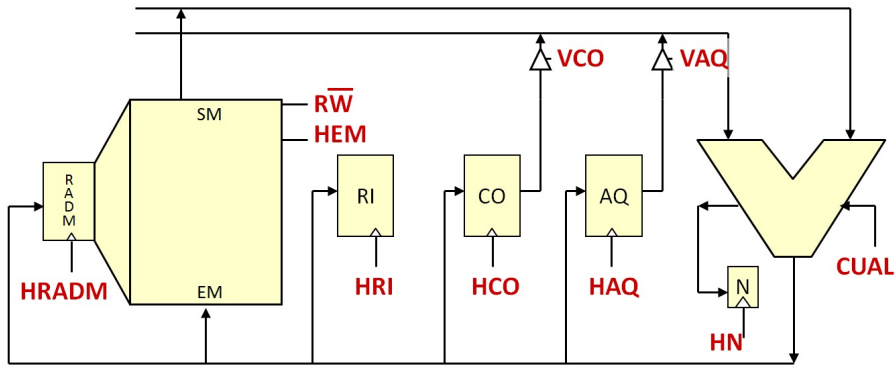
SEQUENCER CLOCK

- Execution of an instruction : several successive steps
- 2 types of signals required :
  - data path configuration signals
  - register loading signals

SEQUENCER CLOCK

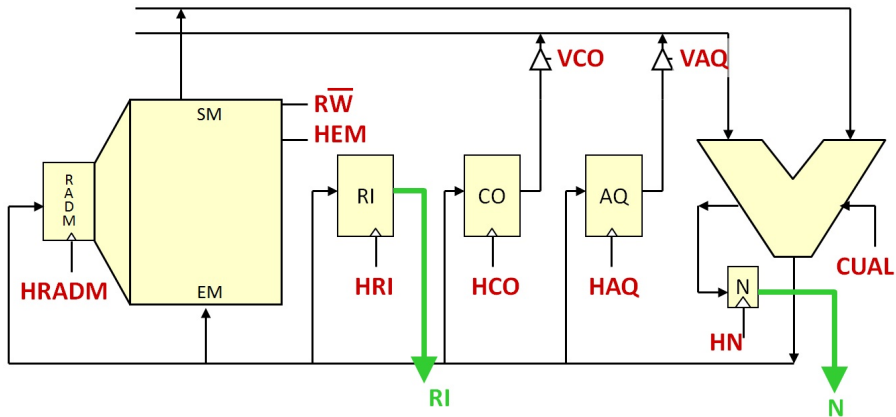


SIGNALS GENERATED BY THE SEQUENCER



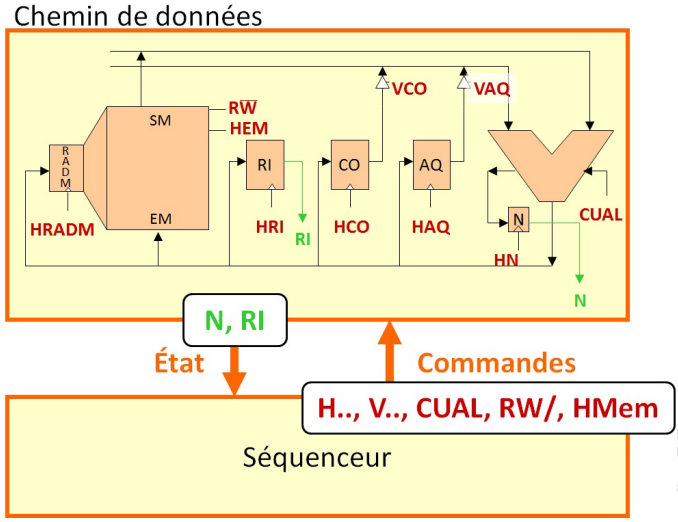
Commandes

SIGNALS TO PROVIDE TO THE SEQUENCER



Signaux d'état

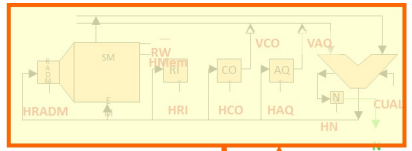
SEQUENCER AND DATA PATH



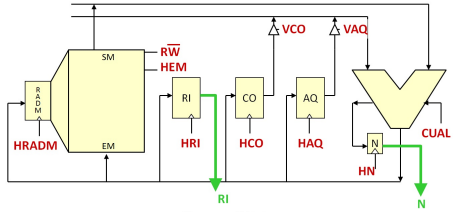
IMPLEMENTING THE SEQUENCER

Each time, generate the control signals according to the status signals

Chemin de données

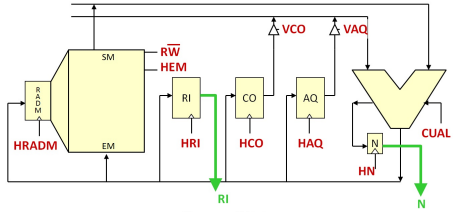


Séquenceur



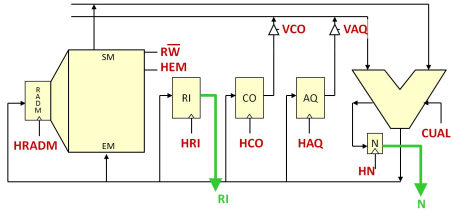
Signaux d'état

LDA	T1	T2	T3	T4	T5	T6	T7	T8
Imm								
Dir								
Ind								



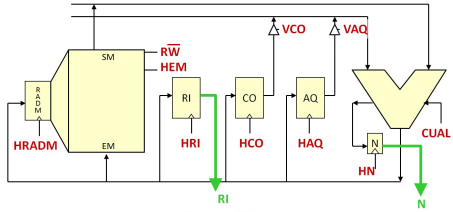
Signaux d'état

LDA	T1	T2	T3	T4	T5	T6	T7	T8
Imm								
Dir								
Ind	VCO CUAL=V1 HRADM	VCO CUAL=V1+1 HCO	CUAL=V2 HRI	VCO CUAL=V1 HRADM	VCO CUAL=V1+1 HCO	CUAL=V2 HRADM	CUAL=V2 HRADM	CUAL=V2 HAQ HN



Signaux d'état

LDA	T1	T2	T3	T4	T5	T6	T7	T8
Imm								
Dir	VCO CUAL=V1 HRADM	VCO CUAL=V1+1 HCO	CUAL=V2 HRI	VCO CUAL=V1 HRADM	VCO CUAL=V1+1 HCO	CUAL=V2 HRADM	CUAL=V2 HAQ HN	
Ind	VCO CUAL=V1 HRADM	VCO CUAL=V1+1 HCO	CUAL=V2 HRI	VCO CUAL=V1 HRADM	VCO CUAL=V1+1 HCO	CUAL=V2 HRADM	CUAL=V2 HRADM	CUAL=V2 HAQ HN



Signaux d'état

LDA	T1	T2	T3	T4	T5	T6	T7	T8
Imm	VCO CUAL=V1 HRADM	VCO CUAL=V1+1 HCO	CUAL=V2 HRI	VCO CUAL=V1 HRADM	VCO CUAL=V1+1 HCO	CUAL=V2 HAQ HN		
Dir	VCO CUAL=V1 HRADM	VCO CUAL=V1+1 HCO	CUAL=V2 HRI	VCO CUAL=V1 HRADM	VCO CUAL=V1+1 HCO	CUAL=V2 HRADM	CUAL=V2 HAQ HN	
Ind	VCO CUAL=V1 HRADM	VCO CUAL=V1+1 HCO	CUAL=V2 HRI	VCO CUAL=V1 HRADM	VCO CUAL=V1+1 HCO	CUAL=V2 HRADM	CUAL=V2 HRADM	CUAL=V2 HAQ HN

TIMES TABLE

LDA	T1	T2	T3	T4	T5	T6	T7	T8
Imm	VCO CUAL=V1 HRADM	VCO CUAL=V1+1 HCO	CUAL=V2 HRI	VCO CUAL=V1 HRADM	VCO CUAL=V1+1 HCO	CUAL=V2 HAQ HN		
Dir	idem	idem	idem	idem	idem	CUAL=V2 HRADM	CUAL=V2 HAQ HN	
Ind	idem	idem	idem	idem	idem	idem	CUAL=V2 HRADM	CUAL=V2 HAQ HN

TIMES TABLE

LDA	T1	T2	T3	T4	T5	T6	T7	T8
Imm	VCO CUAL=V1 HRADM	VCO CUAL=V1+1 HCO	CUAL=V2 HRI	VCO CUAL=V1 HRADM	VCO CUAL=V1+1 HCO	CUAL=V2 HAQ HN		
Dir	idem	idem	idem	idem	idem	CUAL=V2 HRADM	CUAL=V2 HAQ HN	
Ind	idem	idem	idem	idem	idem	idem	CUAL=V2 HRADM	CUAL=V2 HAQ HN

CentraleSupélec université PARIS-SACLAY

TIMES TABLE

LDA	T1	T2	T3	T4	T5	T6	T7	T8
Imm	VCO CUAL=V1 HRADM	VCO CUAL=V1+1 HCO	CUAL=V2 HRI	VCO CUAL=V1 HRADM	VCO CUAL=V1+1 HCO			CUAL=V2 HAQ HN
Dir	idem	idem	idem	idem	idem	CUAL=V2 HRADM		CUAL=V2 HAQ HN
Ind	idem	idem	idem	idem	idem	idem	CUAL=V2 HRADM	CUAL=V2 HAQ HN

TIMES TABLE

LDA	T1	T2	T3	T4	T5	T6	T7	T8
Imm	VCO CUAL=V1 HRADM	VCO CUAL=V1+1 HCO	CUAL=V2 HRI	VCO CUAL=V1 HRADM	VCO CUAL=V1+1 HCO			CUAL=V2 HAQ HN
Dir	idem	idem	idem	idem	idem	CUAL=V2 HRADM		CUAL=V2 HAQ HN
Ind	idem	idem	idem	idem	idem	idem	CUAL=V2 HRADM	CUAL=V2 HAQ HN

TIMES TABLE

LDA	T1	T2	T3	T4	T5	T6	T7	T8
Imm	VCO CUAL=V1 HRADM	VCO CUAL=V1+1 HCO	CUAL=V2 HRI	VCO CUAL=V1 HRADM	VCO CUAL=V1+1 HCO			CUAL=V2 HAQ HN
Dir	idem	idem	idem	idem	idem	CUAL=V2 HRADM		CUAL=V2 HAQ HN
Ind	idem	idem	idem	idem	idem	idem	CUAL=V2 HRADM	CUAL=V2 HAQ HN

TIMES TABLE

	T1	T2	T3	T4	T5	T6	T7	T8
LDA	VCO CUAL=V1 HRADM	VCO CUAL=V1+1 HCO	CUAL=V2 HRI	VCO CUAL=V1 HRADM	VCO CUAL=V1+1 HCO	CUAL=V2 HRADM si Dir ou Ind	CUAL=V2 HRADM si Ind	CUAL=V2 HAQ HN

TIMES TABLE

	T1	T2	T3	T4	T5	T6	T7	T8
	VCO CUAL=V1 HRADM	VCO CUAL=V1+1 HCO	CUAL=V2 HRI	VCO CUAL=V1 HRADM	VCO CUAL=V1+1 HCO	CUAL=V2 HRADM si Dir ou Ind	CUAL=V2 HRADM si Ind	...

THANK YOU

Back to the begin - Back to the outline

CentraleSupélec | université PARIS-SACLAY