S34PH412 : Systèmes Microprogrammés

Architecture des Ordinateurs

Cours 3

Université de la Réunion

Année 2015/2016

Alicalapa F.

UNIVERSITÉ DE LA RÉUNION
S'ouvrir aux mondes

UFR Sciences et Technologies UNIVERSITÉ DE LA RÉUNION

Exercice Moodle à faire auto-formation :

Fichiers de CMs - CIs - TDs

Fichiers de CMs - CIs - TDs

Support de CMs/CIs

TD1 2128 impression octets numeration operations

Documents de découverte et quizz

Documents de découverte et quizz

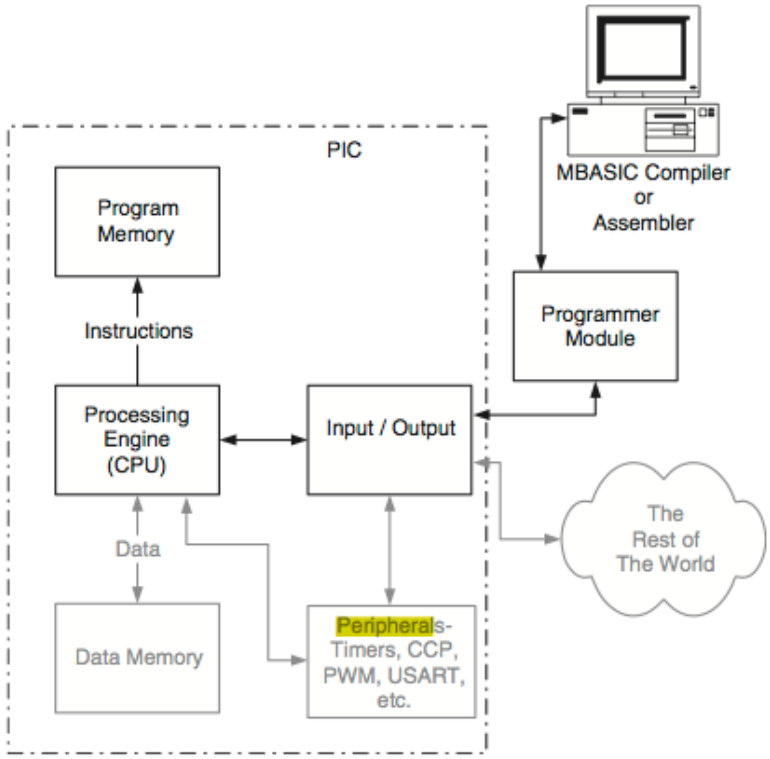
Qu'est-ce qu'un microcontrôleur ?

Convention pour les slides : **éléments à noter, sera effacé.**

Concepts du cours/TD n°2 précédent :

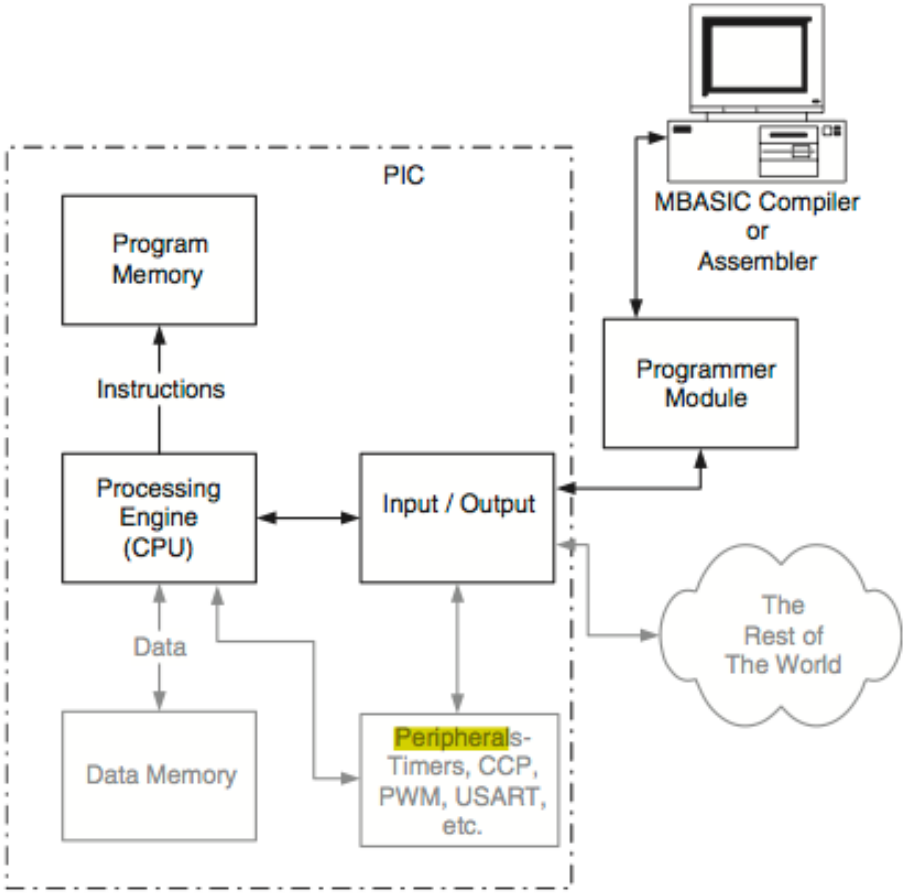
- Notion de CPU et ses fonctions de logique booléenne
- Notions d'horloges, rôle, défauts et idéalités, Shannon et les multiples horloges
- La mémoire centrale … utilité pour les calculs de la fonction mémoire
- Les périphériques du uC

Device number	No. of Pins*	Clock Speed	Memory (K = Kbytes, i.e. 1024 bytes)	Peripherals/Special Features
16F84A	18	DC to 20 MHz	1K program memory, 68 bytes RAM, 64 bytes EEPROM	1 8-bit timer 1 5-bit parallel port 1 8-bit parallel port
16LF84A	As above	As above	As above	As above, with extended supply voltage range
16F84A-04	As above	DC to 4 MHz	As above	As above
16F873A	28	DC to 20 MHz	4K program memory 192 bytes RAM, 128 bytes EEPROM	3 parallel ports, 3 counter/timers, 2 capture/compare/PWM modules, 2 serial communication modules, 5 10-bit ADC channels, 2 analog comparators



Concepts du cours/TD n°2 précédent :

- **4 grandes fonctions : CPU + HORLOGE + PERIPHERIQUES + MEMOIRE**



- **Binaire, hexa, octal**
- **Calculs, -, +**
- **Notation IEEE 754**

- IEEE 754 - **Multiplication de deux nombres flottants**
  - Multiplier les mantisses.
  - Additionner les exposants en soustrayant une fois le biais pour retomber sur un exposant biaisé.
  - Si nécessaire, renormaliser la mantisse sous la forme 1,*M* et décaler l'exposant.
  - Arrondir la mantisse (car la multiplication a généré plus de bits significatifs).
  - Ajuster le signe du résultat en fonction des signes des nombres à multiplier.
- Exemple : $1.75_{10} * 2.5_{10}$ … pour 2.5 remarquons $2.5=1.25*2^1$

$1{,}75 = 1{,}75 \times 2^0$, qui se représente ainsi : 0 01111111 11000000000000000000000.

$2{,}5 = 1{,}25 \times 2^1$, qui se représente ainsi : 0 10000000 01000000000000000000000.

- IEEE 754 - **Multiplication de deux nombres flottants**
- Exemple : $1.75_{10} * 2.5_{10}$

$1{,}75 = 1{,}75 \times 2^0$, qui se représente ainsi : 0 01111111 11000000000000000000000.

$2{,}5 = 1{,}25 \times 2^1$, qui se représente ainsi : 0 10000000 01000000000000000000000.

- La multiplication des mantisses (il faut multiplier $1{,}11_2$ par $1{,}01_2$, et pas uniquement les pseudo-mantisses) donne $10{,}0011_2$ et l'addition des exposants aboutit à un exposant biaisé de 128... Auquel on associe 2^1.
- Soit donc $1.00011 * 2^1 * 2^1$), pour renormaliser le nombre en $1{,}00011_2 * 2^2$, qui s'écrit alors 0 10000001 00011000000000000000000. Cela donne bien $(1 + 2^{-4} + 2^{-5}) * 2^2 = 4{,}375$.
- La division flottante s'effectue de façon semblable : on divise les mantisses et on soustrait les exposants.

129 biaisé

- IEEE 754 - **Addition de deux nombres flottants**

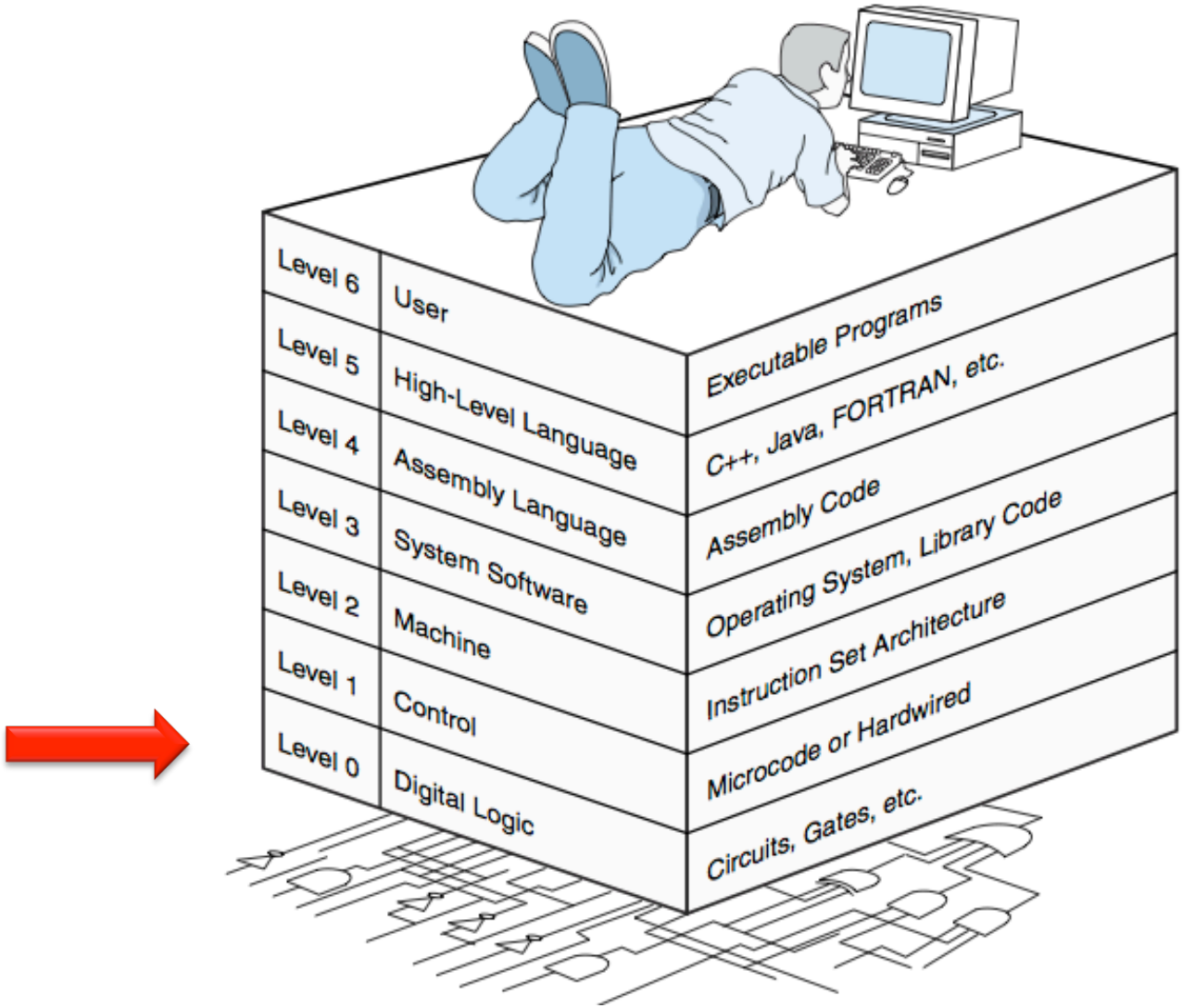
- L'addition de deux nombres flottants ne peut se faire que si les exposants sont égaux. L'algorithme est le suivant :

1. Décaler la mantisse d'un des nombres pour aligner les exposants.
2. Additionner les mantisses (attention au signe).
3. Si nécessaire, renormaliser la mantisse sous la forme 1, M et décaler l'exposant.
4. Arrondir la mantisse (car, à cause du décalage initial, la mantisse peut être sur plus de bits que la taille autorisée).

dépassement

- IEEE 754 - **Addition de deux nombres flottants**
- Exemple : $1.75_{10} + 2.5_{10}$
- 1.75_{10} -> $1{,}11_2 * 2^0$ et 2.5_{10} -> $1{,}01_2 * 2^1$
- décale le premier nombre en l'écrivant $0{,}111_2 * 2^1$ pour aligner les exposants sur 2^1.
- L'addition des deux mantisses donne $1{,}01_2 + 0{,}111_2 = 10{,}001_2$.
- On normalise en décalant vers la droite et en ajoutant 1 à l'exposant, ce qui amène à un résultat : $1{,}0001_2 * 2^2$, soit 4,25.
- On réalise la soustraction de façon identique, en établissant la différence les mantisses.

- Circuits logiques de l'**Addition … quid ?**



- Circuits logiques de l'**Addition**

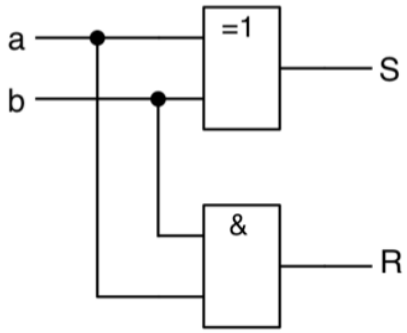
- En se basant sur l'algèbre de Boole (rappel ??) nous pouvons considérer le tableau suivant qui décrit le **Demi-additionneur** :

a	b	S	R
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

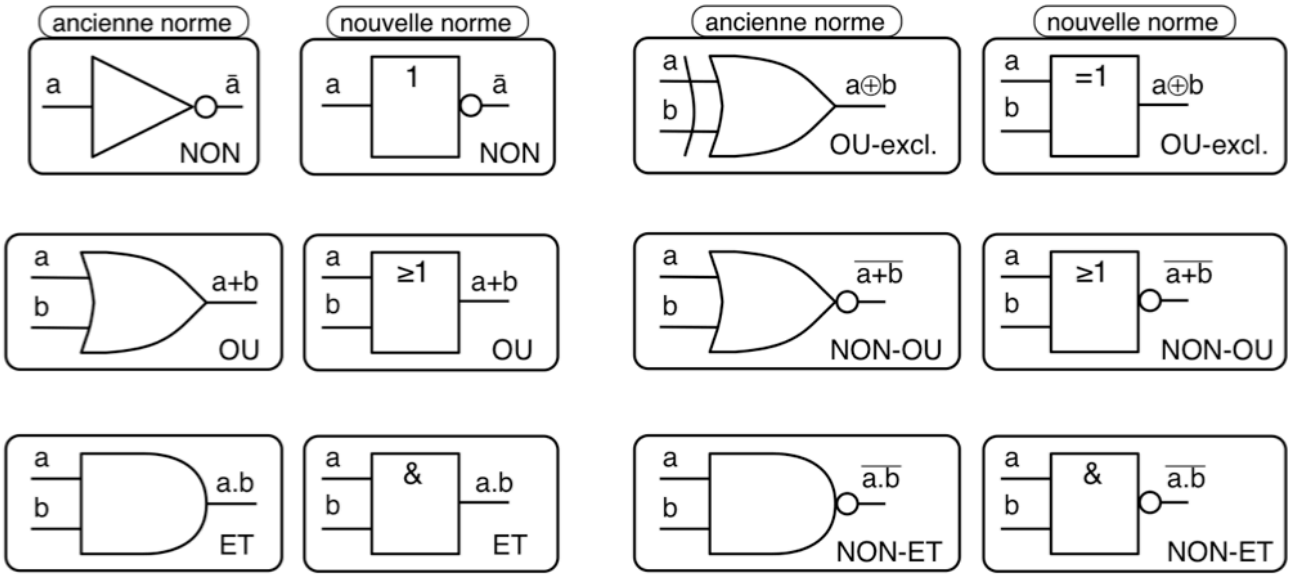
Addition de 2 bits

- Circuits logiques de l'**Addition**

- le Demi-additionneur :



Demi-additionneur



- Circuits logiques de l'**Addition**

- Par extension nous devons considérer, vu nos calculs de TD, l'additionneur complet (avec retenue):

a	b	r	S	R'
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

- Circuits logiques de l'**Addition**

- Par extension nous devons considérer, vu nos calculs de TD, l'additionneur complet (avec retenue):

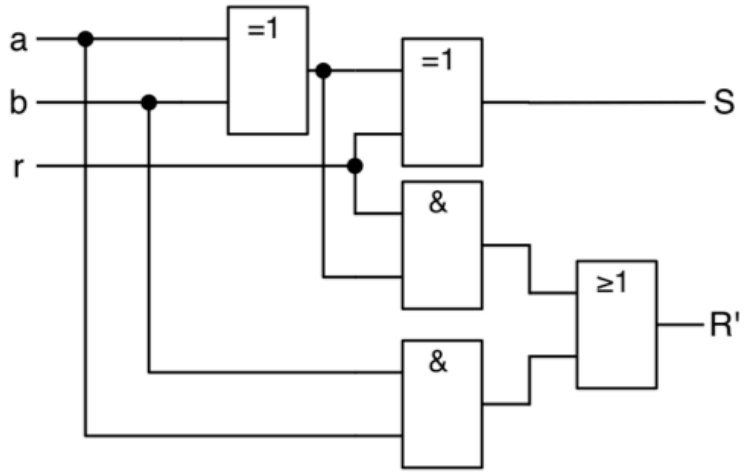
a	b	r	S	R'
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

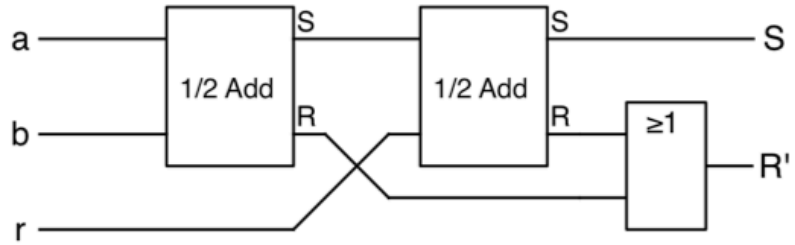
$$S = a \oplus b \oplus r$$

$$\begin{aligned} R' &= \bar{a}br + a\bar{b}r + ab\bar{r} + abr \\ &= r(\bar{a}b + a\bar{b}) + ab(\bar{r} + r) \\ &= r(a \oplus b) + ab \end{aligned}$$

- Circuits logiques de l'**Addition**

- Par extension nous devons considérer, vu nos calculs de TD, l'additionneur complet (avec retenue):





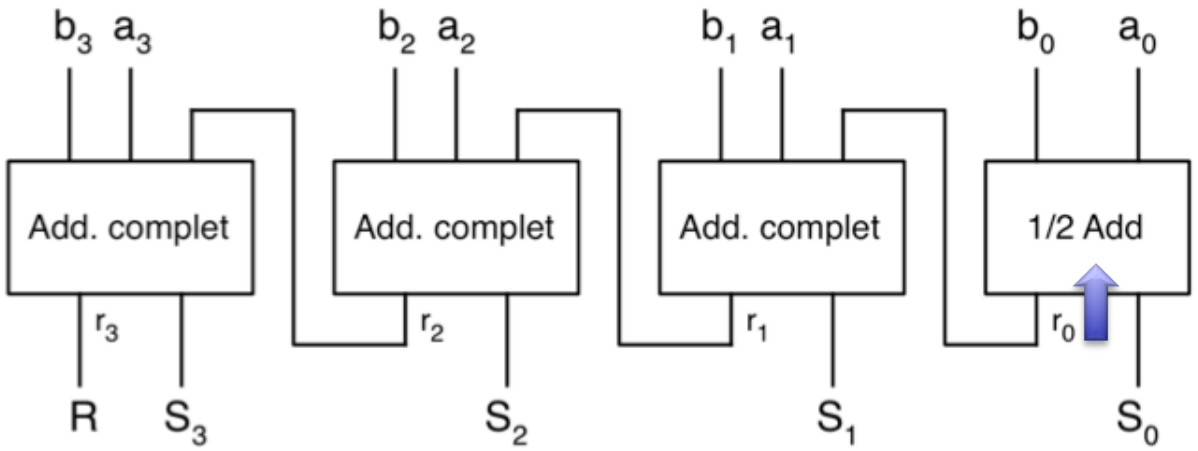
Additionneur complet

$$S = a \oplus b \oplus r$$

$$\begin{aligned} R' &= \bar{a}br + a\bar{b}r + ab\bar{r} + abr \\ &= r(\bar{a}b + a\bar{b}) + ab(\bar{r} + r) \\ &= r(a \oplus b) + ab \end{aligned}$$

- Circuits logiques de l'**Addition**

- Par extension additionneur 4 bits :



Additionneur 4 bits.

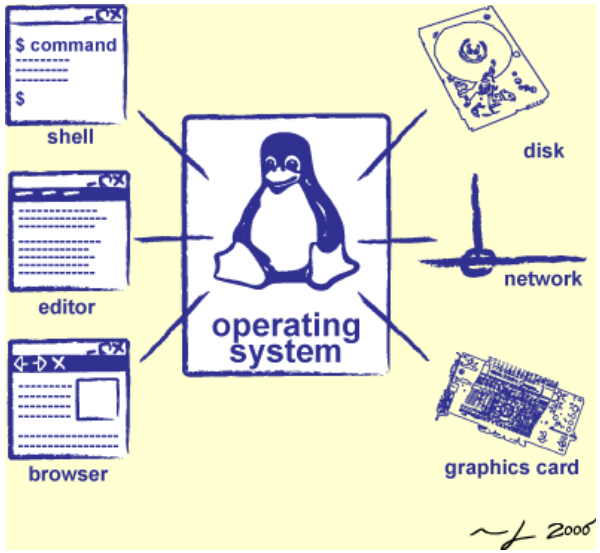
I. Notion de numération binaire (le monde du numérique et de l'analogique)

a. Vocabulaire de base du numérique : bit, byte, Ko, octet
b. Domaines numérique et analogique
c. Numération : utilité ?
d. Le codage binaire, Binaire naturel
e. D'autres bases existent – octal et hexadécimal
f. Binaire réfléchi, code de Gray
g. Autres codes
h. Représentation des nombres signés en binaire
i. nombres flottants ou à virgule en base 2
j. Hardware - software

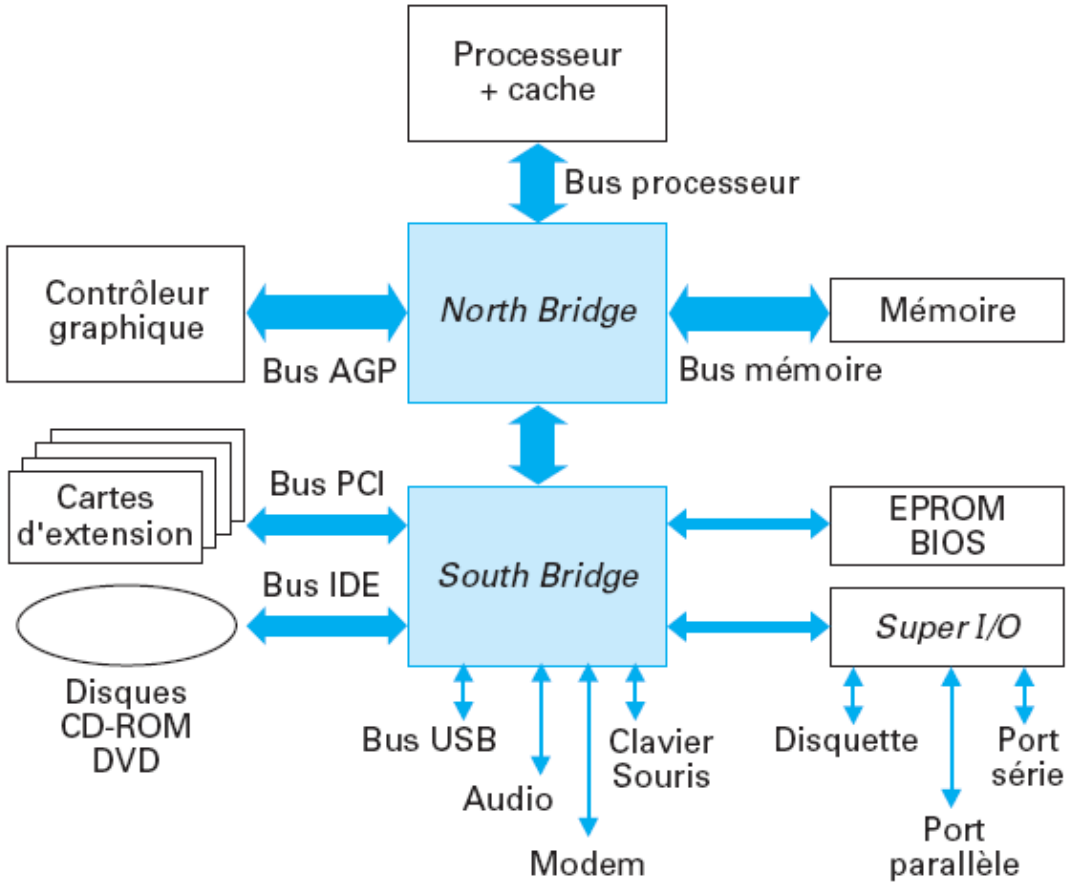
II. Rapide Historique de l'évolution des ordinateurs

Les éléments de base d'un ordinateur

- Au niveau des ordinateurs nous avons 2 grands domaines :
  - **Hardware** = disque dur, lecteur DVD, carte son, carte vidéo
  - Et le **software** = OS(operating system), logiciel de traitement de texte, de traitement vidéo….
- Notre **interaction** avec l'ordinateur = essentiellement lié au **hardware** (clavier, écran, carte graphique, carte son…)
- Notre mode de **raisonnement** est plus lié au **software**
- Pour **« programmer » le hardware** et réaliser des actions et des fonctions avec eux, nous avons **besoin d'un OS** qui va s'occuper du **dialogue** avec eux et de leur **gestion**.

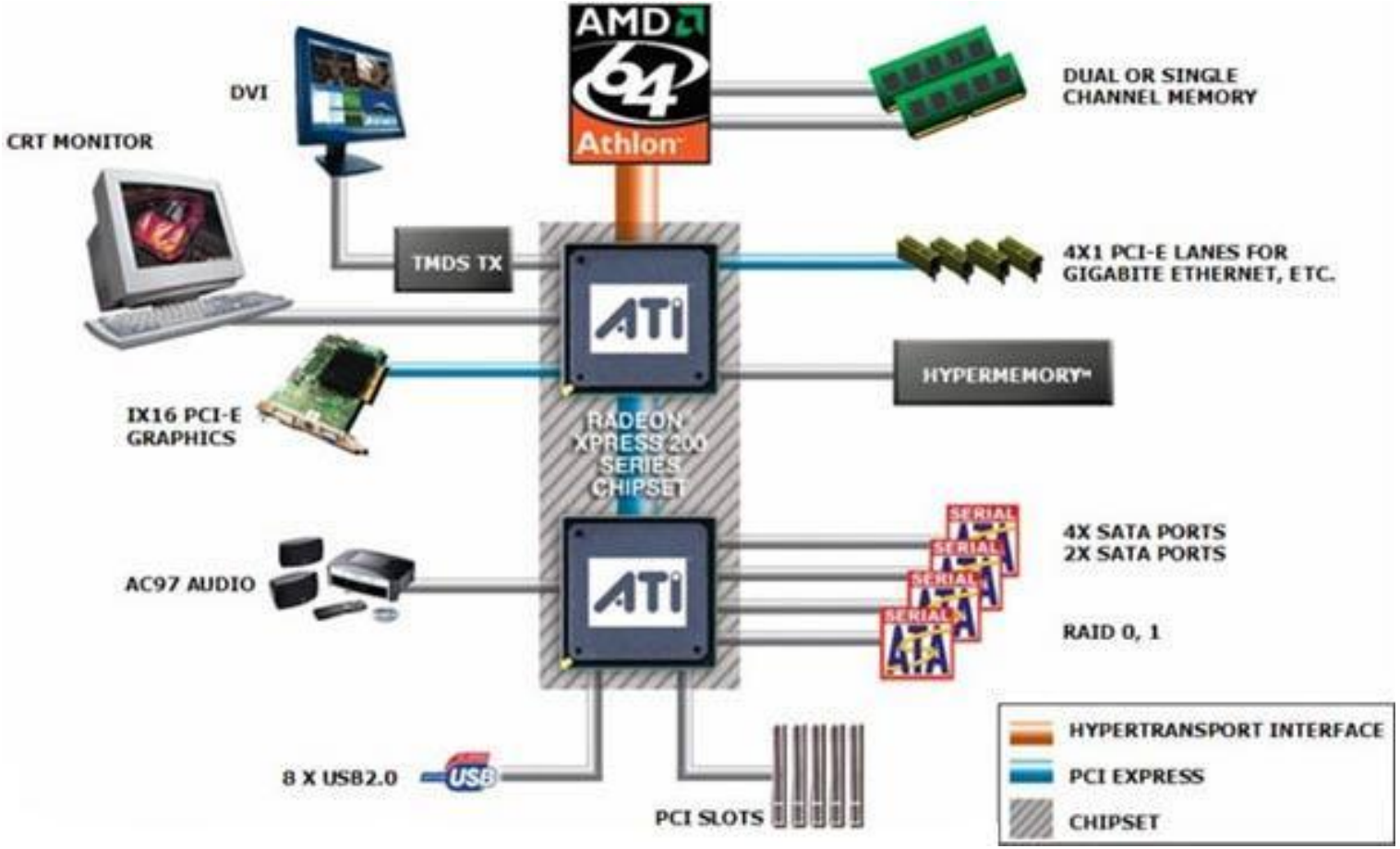


- Exemple de vision de l'hardware d'un PC :
  - Pont Nord et Sud

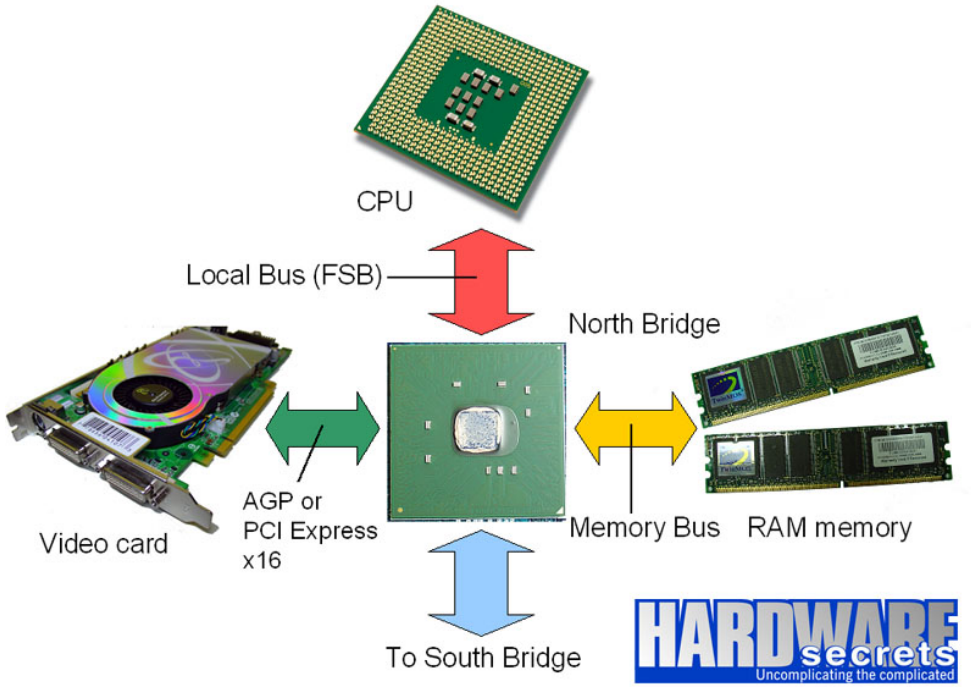


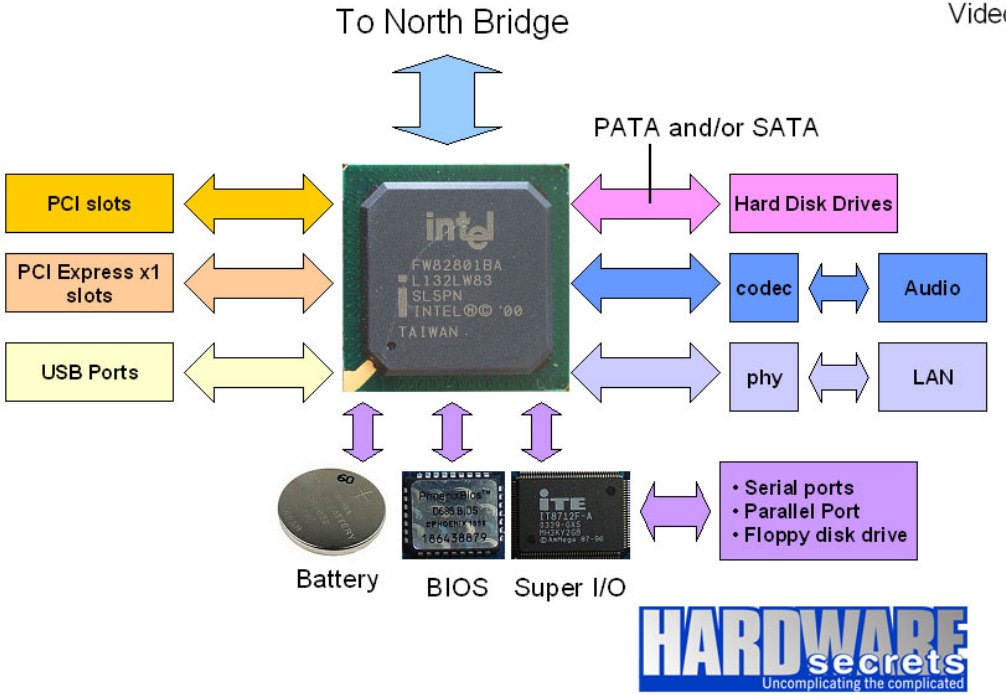
culture

- chipset


AMD 64 Athlon
DVI
CRT MONITOR
TMDS TX
DUAL OR SINGLE CHANNEL MEMORY
4X1 PCI-E LANES FOR GIGABITE ETHERNET, ETC.
HYPERMEMORY™
IX16 PCI-E GRAPHICS
RADEON XPRESS 200 SERIES CHIPSET
ATI
ATI
AC97 AUDIO
4X SATA PORTS
2X SATA PORTS
RAID 0, 1
SERIAL ATA
8 X USB2.0
USB
PCI SLOTS
HYPERTRANSPORT INTERFACE
PCI EXPRESS
CHIPSET

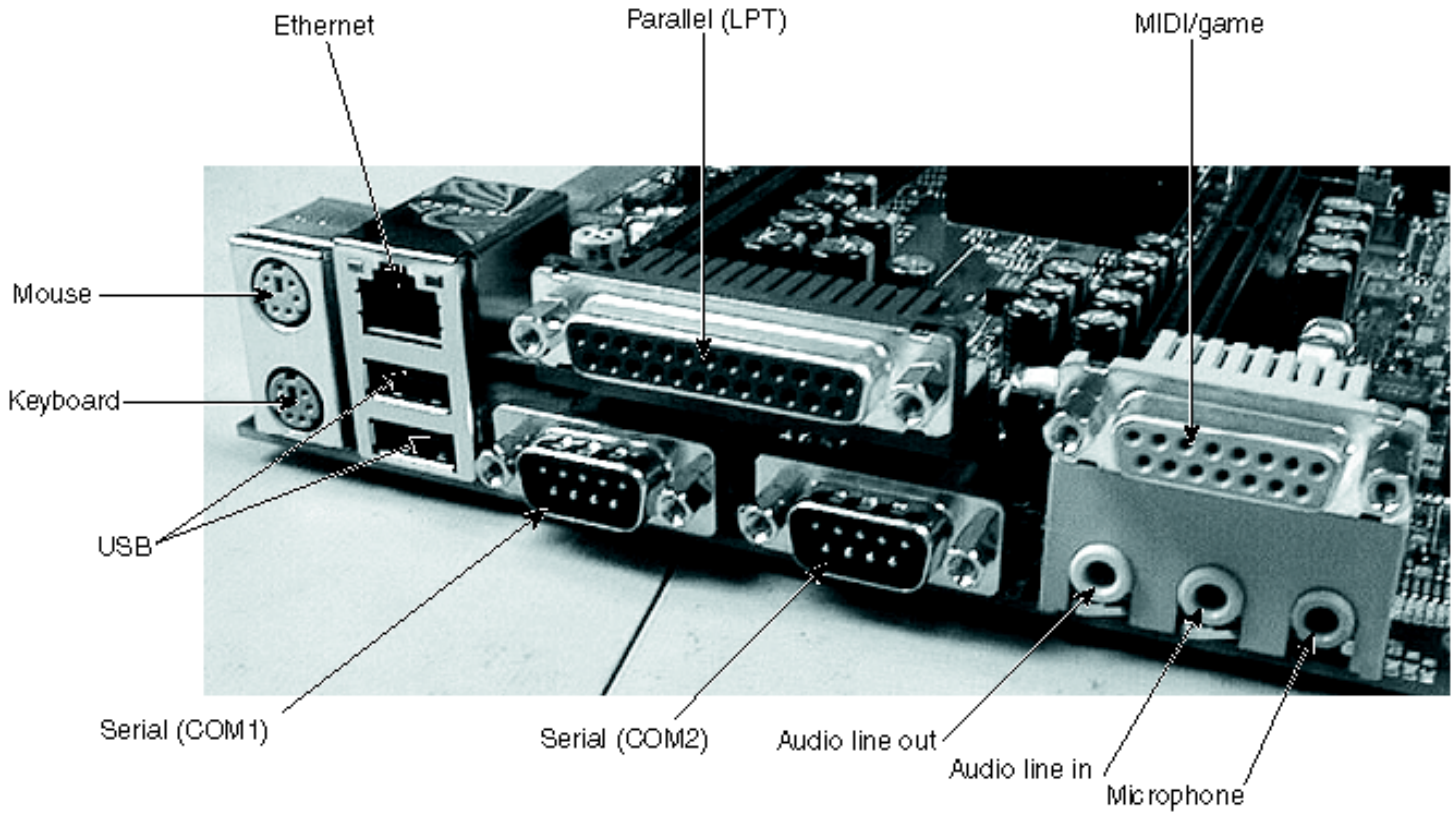
- chipset

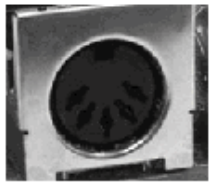
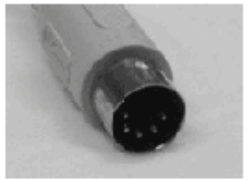
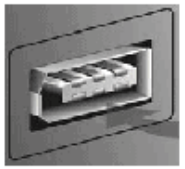
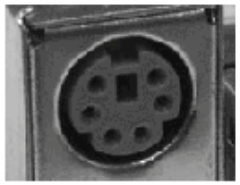
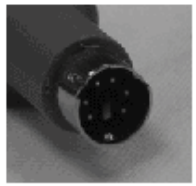
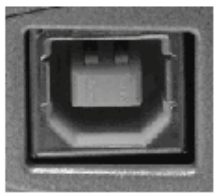
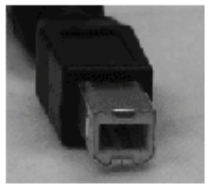

CPU
Local Bus (FSB)
North Bridge
Video card
AGP or
PCI Express
x16
Memory Bus
RAM memory
To South Bridge
HARDWARE secrets
Uncomplicating the complicated


To North Bridge
PATA and/or SATA
PCI slots
PCI Express x1 slots
USB Ports
Hard Disk Drives
codec
Audio
phy
LAN
• Serial ports
• Parallel Port
• Floppy disk drive
Battery
BIOS
Super I/O
HARDWARE secrets
Uncomplicating the complicated

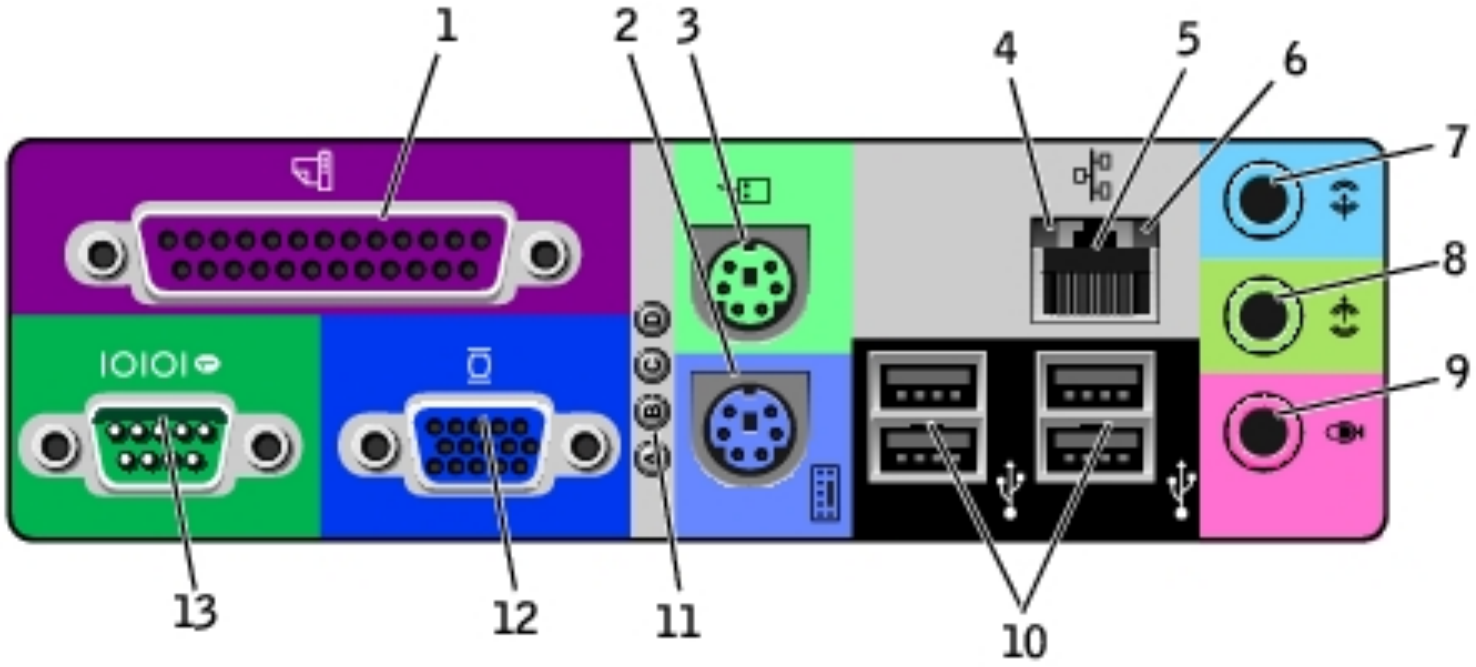
Les éléments de base d'un ordinateur

- Connecteurs arrières (à connaître) : certains se retrouvent sur les cartes de **systèmes embarqués**

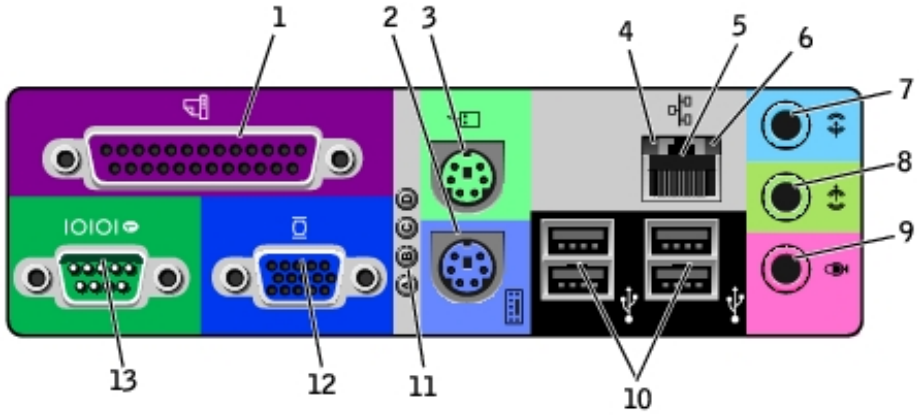


Connector	Common Uses	Device Interface	Cable Connector
DIN-5	AT Keyboard		
FireWire (IEEE-1394)	High-Bandwidth Devices		
MiniDIN-6 (PS/2)	Keyboard, Mouse		
MiniRCA	Speakers, Microphone		
USB A	USB 1.1 and USB 2.0		
USB B	USB 1.1 and USB 2.0 (detachable devices)		

- Questions ? Qui est quoi ? (réponse après)



- Questions ? Qui est quoi ?



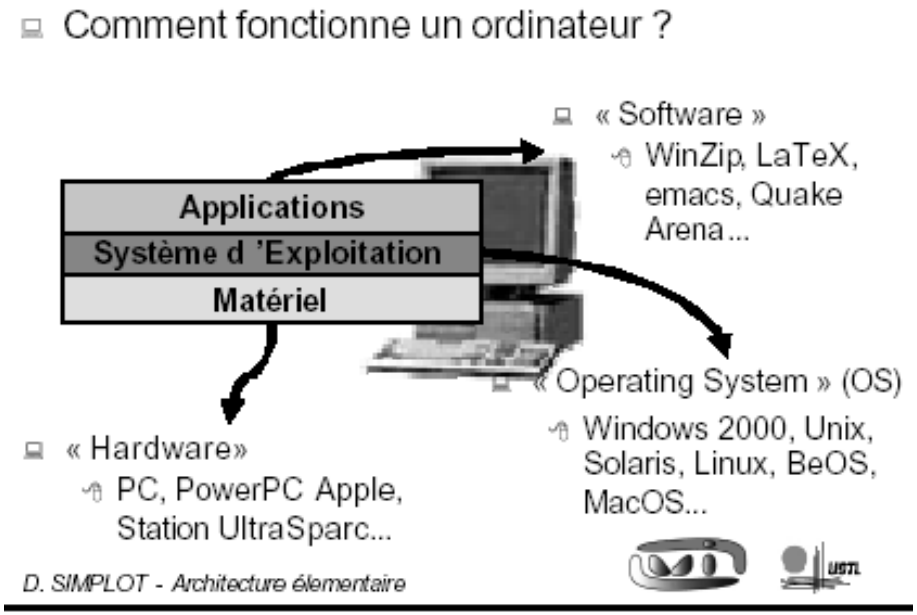
- GPIB

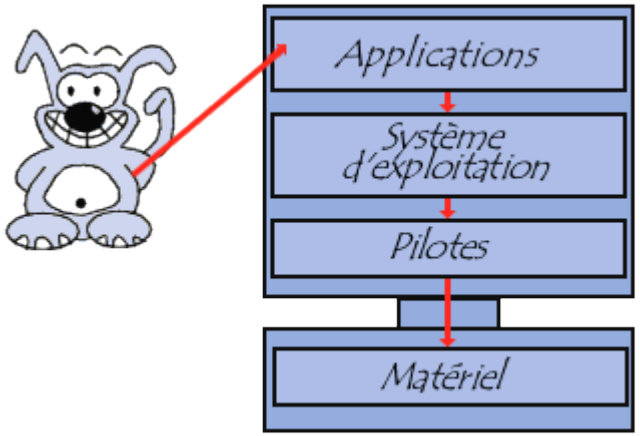
1	parallel connector
2	keyboard connector
3	mouse connector
4	link integrity light
5	network adapter
6	network activity light
7	line-in connector
8	line-out connector
9	microphone connector
10	USB connectors
11	diagnostic lights
12	video connector
13	serial connector

- GPIB


PORT
CENTRONICS PORT
CAUTION
ATTENTION STATIC SENSITIVE DEVICES HANDLE ONLY AT A STATIC SAFE WORK STATION
GPIB IEEE 488
Tektronix
P.S SUP MOD 40-003
CAT II

- Pour qu'un programme puisse s'exécuter, le système doit pouvoir assurer la communication entre les différents éléments et leur disponibilité (µP, mémoire, carte video…)…c'est le rôle du système d'exploitation (OS).
- Gérer les ressources matériels et proposer à l'utilisateur des techniques simples pour « s'adresser » aux éléments souhaités.





- Le noyau de l'OS (ou cœur, kernel) assure les fonctionnalités suivantes :

1. **Affecter ….**
2. **Synchroniser ..**
3. **Gestion des périphériques** (au moyen de pilotes) ; Gérer les opérations d'entrée-sortie : affichage à l'écran, saisie au clavier et à la souris, lecture-écriture sur des disques, impression, etc.
4. **Piloter l'accès aux fichiers**.
5. **Permettre l'accès aux réseaux**.

- Les OSs sont capables de réaliser "presque" en même temps ces différentes actions : on les qualifie **de multitâches**.

I. Notion de numération binaire (le monde du numérique et de l'analogique)

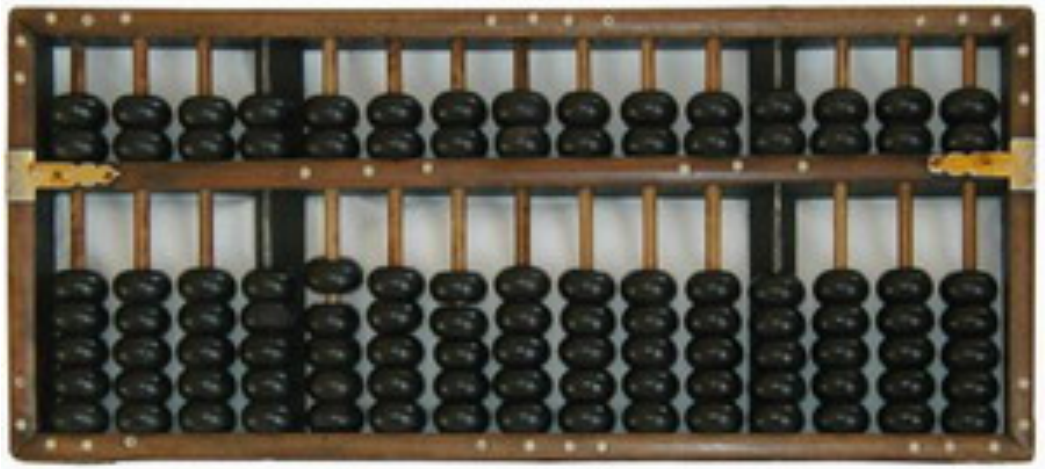
II. Rapide Historique de l'évolution des ordinateurs

III. Le modèle Von Neuman et représentation hiérarchique

IV. Les différents composants du système et leurs caractéristiques

Positionnement historique de l'électronique numérique

Manuel	**Mécanique**	**Electromécanique**	**Électrique**	**Électronique**	**Microélectronique**
Boulier, Abaques, réglette	Roues	Relais	Tube à vide	Transistor	Transistor intégré



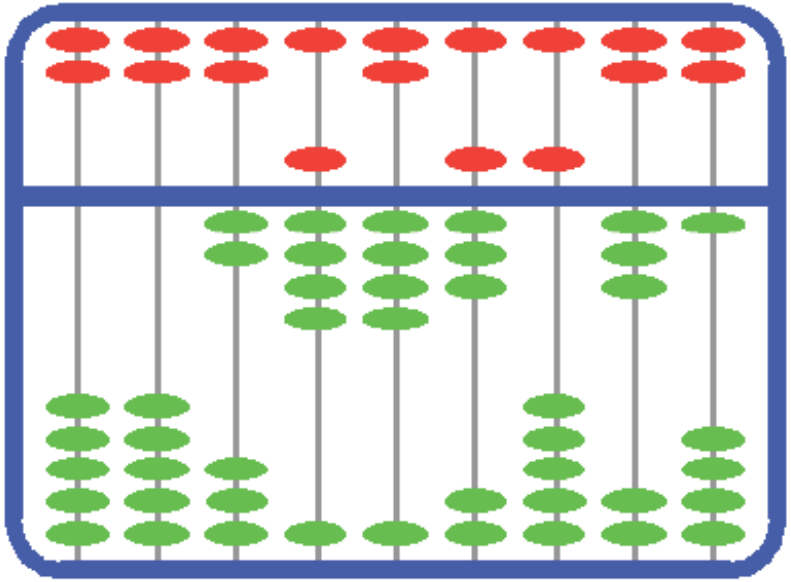
Le Boulier (lien html)

Positionnement historique de l'électronique numérique

Manuel Boulier, Abaques, réglette	**Mécanique** Roues	**Electromécanique** Relais	**Électrique** Tube à vide	**Électronique** Transistor	**Microélectronique** Transistor intégré

Bille rouge 5, bille verte 1, numération positionnelle

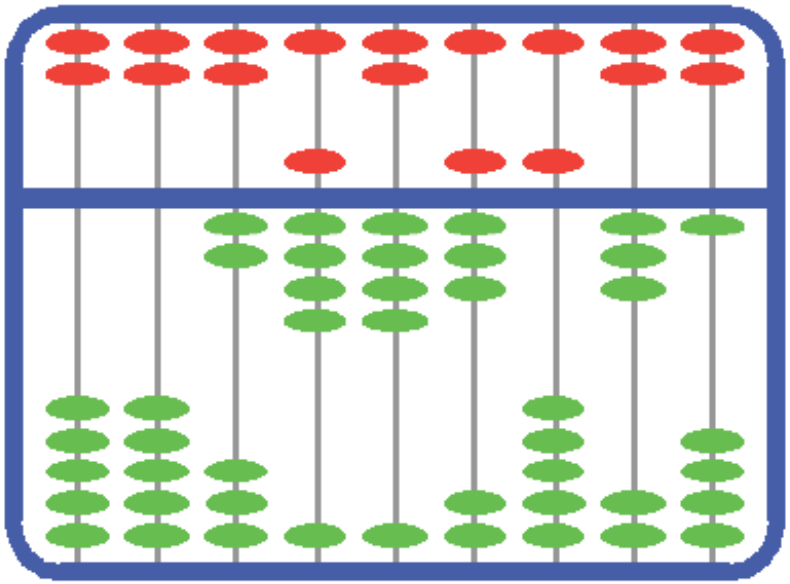
VALEUR ????



Positionnement historique de l'électronique numérique

Manuel	Mécanique	Electromécanique	Électrique	Électronique	Microélectronique
Boulier, Abaques, réglette	Roues	Relais	Tube à vide	Transistor	Transistor intégré

VALEUR ????



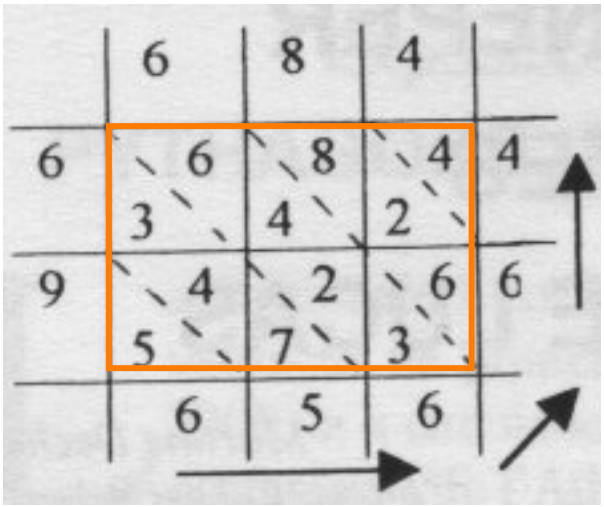
2 948 531 =
(0); (0); (2); (5+4); (4); (5+3); (5+0); (3); (1);

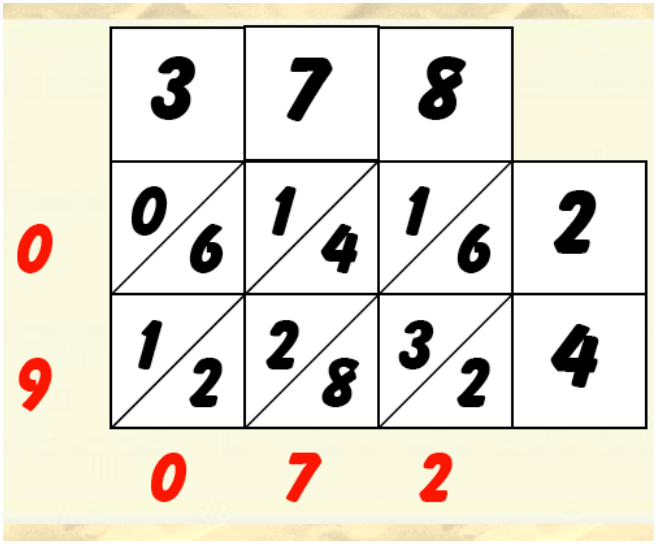
Addition de deux nombres

Exemple: 2 948 531 + 2566 = ?

Positionnement historique de l'électronique numérique

Manuel	Mécanique	Electromécanique	Électrique	Électronique	Microélectronique
Boulier, Abaques, réglette	Roues	Relais	Tube à vide	Transistor	Transistor intégré





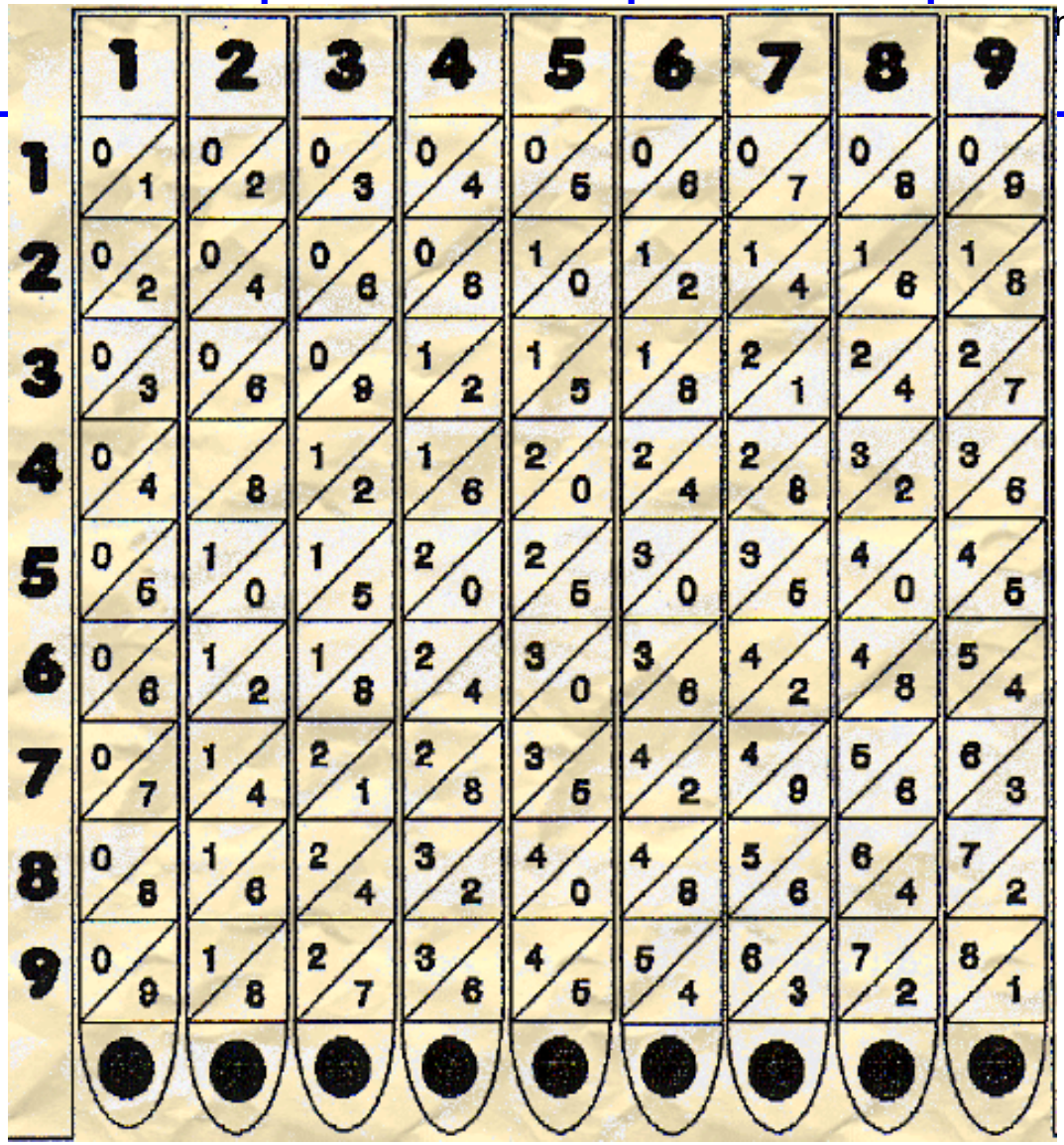
Multiplication musulmane

684*96

- chiffre dizaine en bas de de chaque case.
- addition de chaque bande oblique

Positionnement historique de l'électronique numérique

Manuel	Mécanique	Electromécanique	Électrique	Électronique	Microélectronique
Boulier, Abaques, réglette				nsistor	Transistor intégré

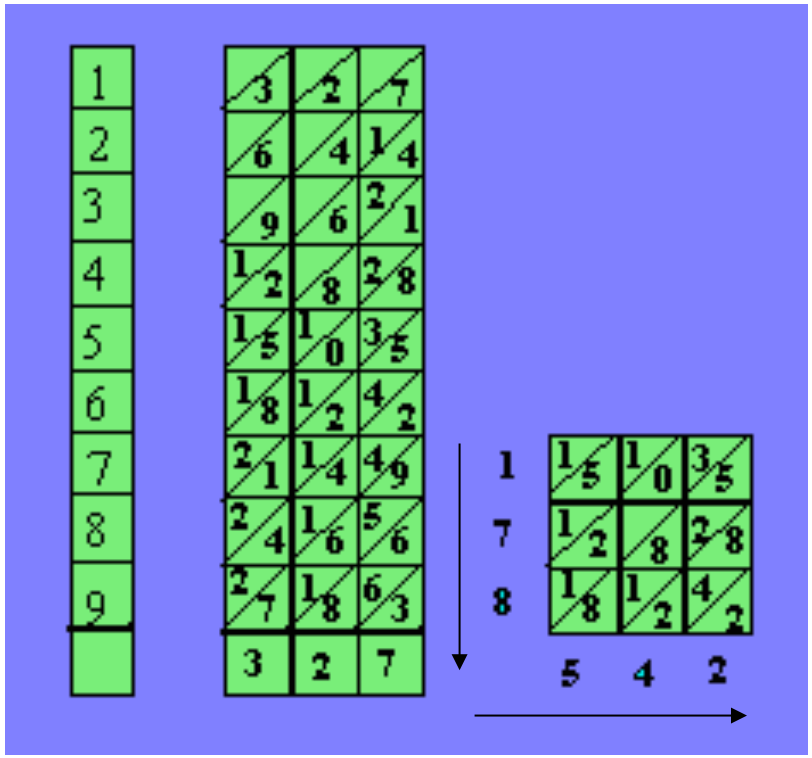


1 - Expliquez comment sont remplies les cases du tableau

2- réaliser : 342*4

Positionnement historique de l'électronique numérique

Manuel	Mécanique	Electromécanique	Électrique	Électronique	Microélectronique
Boulier, Abaques, réglette	Roues	Relais	Tube à vide	Transistor	Transistor intégré



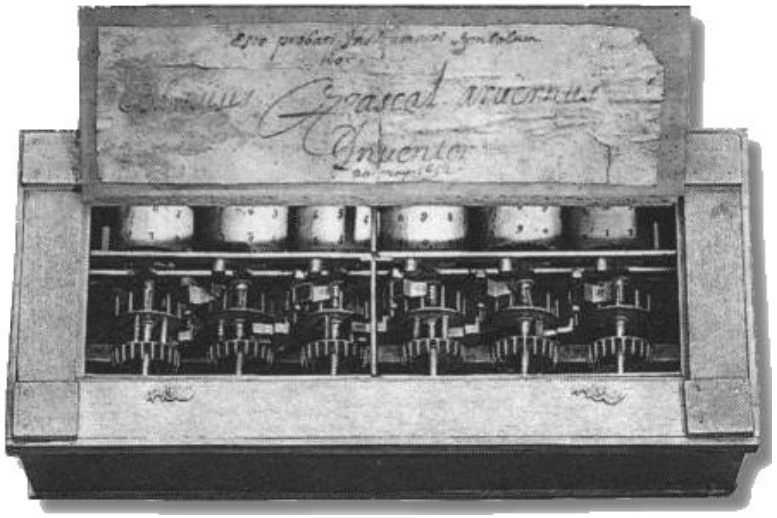
Conclusion de ces 3 cas :
« Savoir bien multiplier revient à savoir bien additionner »

Bâtons de Neper : multiplication de 327 et de 546

- chiffre des dizaines en haut de chaque case
- Bande oblique dans l'autre sens

Positionnement historique de l'électronique numérique

Manuel	Mécanique	Electromécanique	Électrique	Électronique	Microélectronique
Boulier, Abaques, réglette	Roues	Relais	Tube à vide	Transistor	Transistor intégré

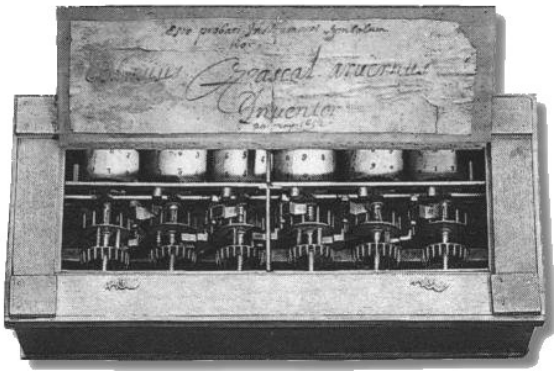


Sur le même principe de calcul :
La Pascaline inventée par Blaise Pascal !
Roues codeuse + notion de décalage des retenues

quelques dates importantes :

- 1617 : Machine à faire les additions et les multiplications par Neper (**Bâtons de Neper**)

- 1642 : Blaise Pascal, « la **Pascaline** », machine à réaliser addition et soustraction à 6 chiffres (utilisation de roues codeuses – maîtrise mécanique)

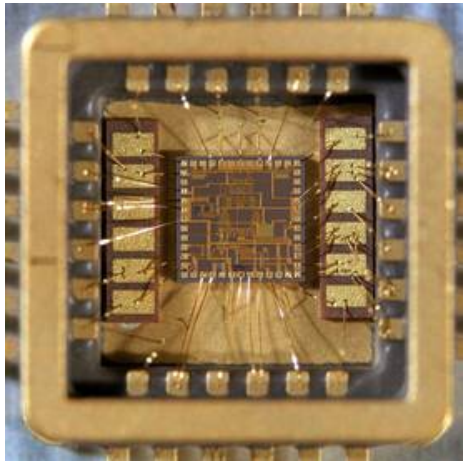


Positionnement historique de l'électronique numérique

Manuel	**Mécanique**	**Electromécanique**	**Électrique**	**Électronique**	**Microélectronique**
Boulier, Abaques, réglette	Roues	Relais	Tube à vide	Transistor	Transistor intégré

Tube à vide
Lampes

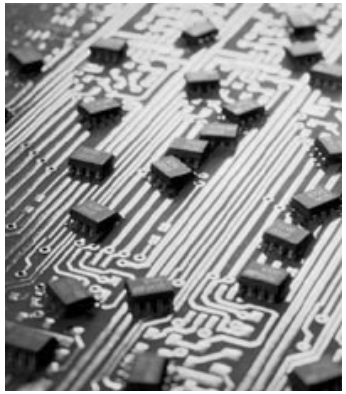
1er transistor



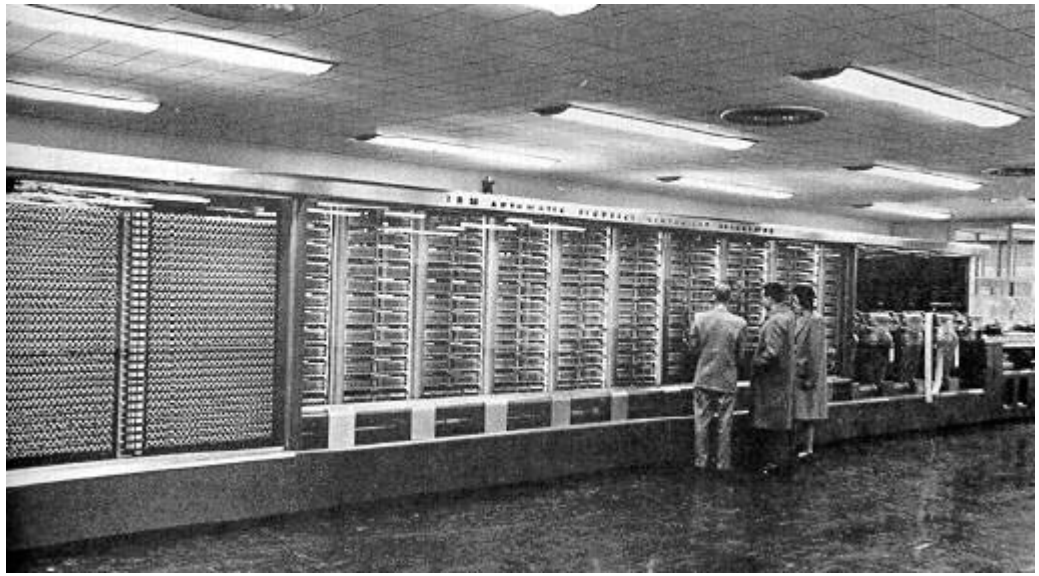
« Circuit intégré »
Circuit à composants intégrés sur du silicium

- 1904 : Invention de la diode (Premier tube à vide) par John Fleming

- 1919 : Invention du basculeur (flip-flop) par W. Eccles et W. Jordan
- 1940 : Invention du circuit imprimé (plaquette comportant des pistes pour relier les composants)

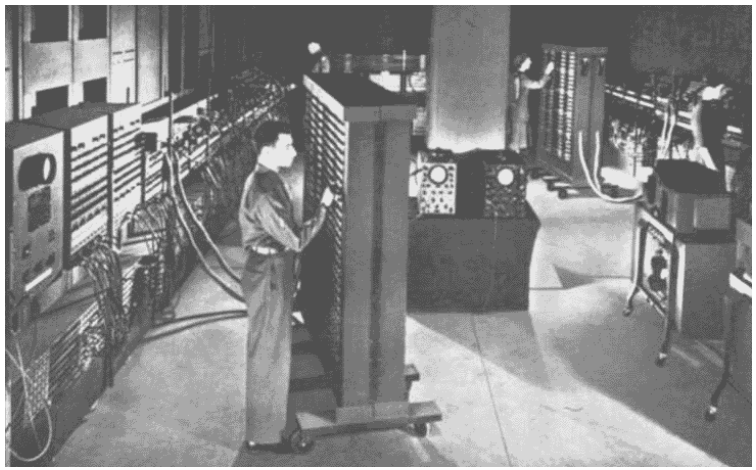


- 1941 : IBM et l'université d'Harvard (H. Aiken), réalisation du **Mark1**, machine à réaliser des calculs (multiplication de 2 nombres de 23 chiffres en 6 secondes), fabriquée à partir de relais et pièces mécaniques



5 tonnes, 25m², Consommation 25 000W à comparer aux alimentations 300Watts des PCs actuels et leur puissance de calculs

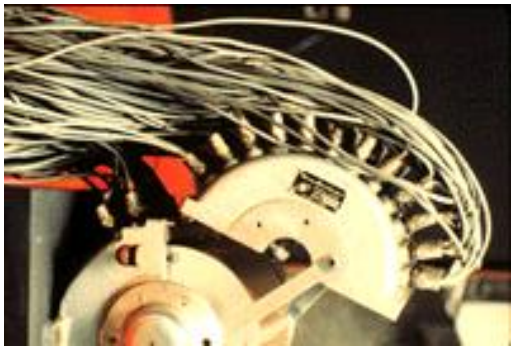
- 1945, réalisation de l'ENIAC (par Mauchly et Eckert) – considéré comme le 1er ordinateur, multiplication de 2 nombres de 10 chiffres en 3ms, réalisé à partir de tubes à vides



- Fin 1945 : John Von Neuman associé à l'équipe de l'ENIAC propose 1er modèle d'ordinateur « l'architecture Von Neuman »

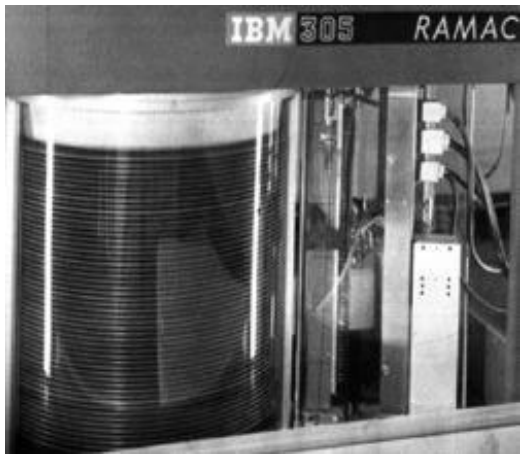
- 1947 : invention du transistor par W. Bradford, W. Brattain et J. Bardeen, dans les laboratoires Bell Telephone

- Années 1950 : mise au point des premières mémoires de masse (ancêtres des disques durs)

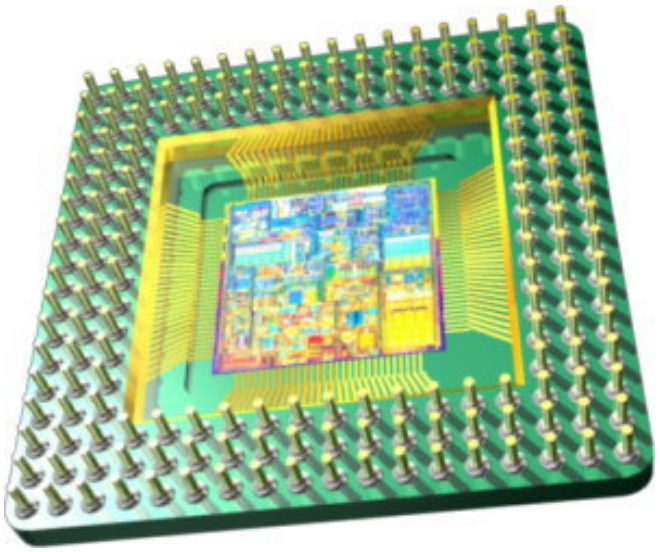


Tambour de masse magnétique ERA 1101

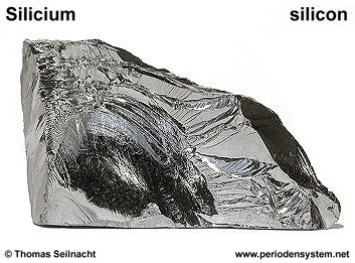
RAMAC305 (5Mo, 50 disques de 61 cm)



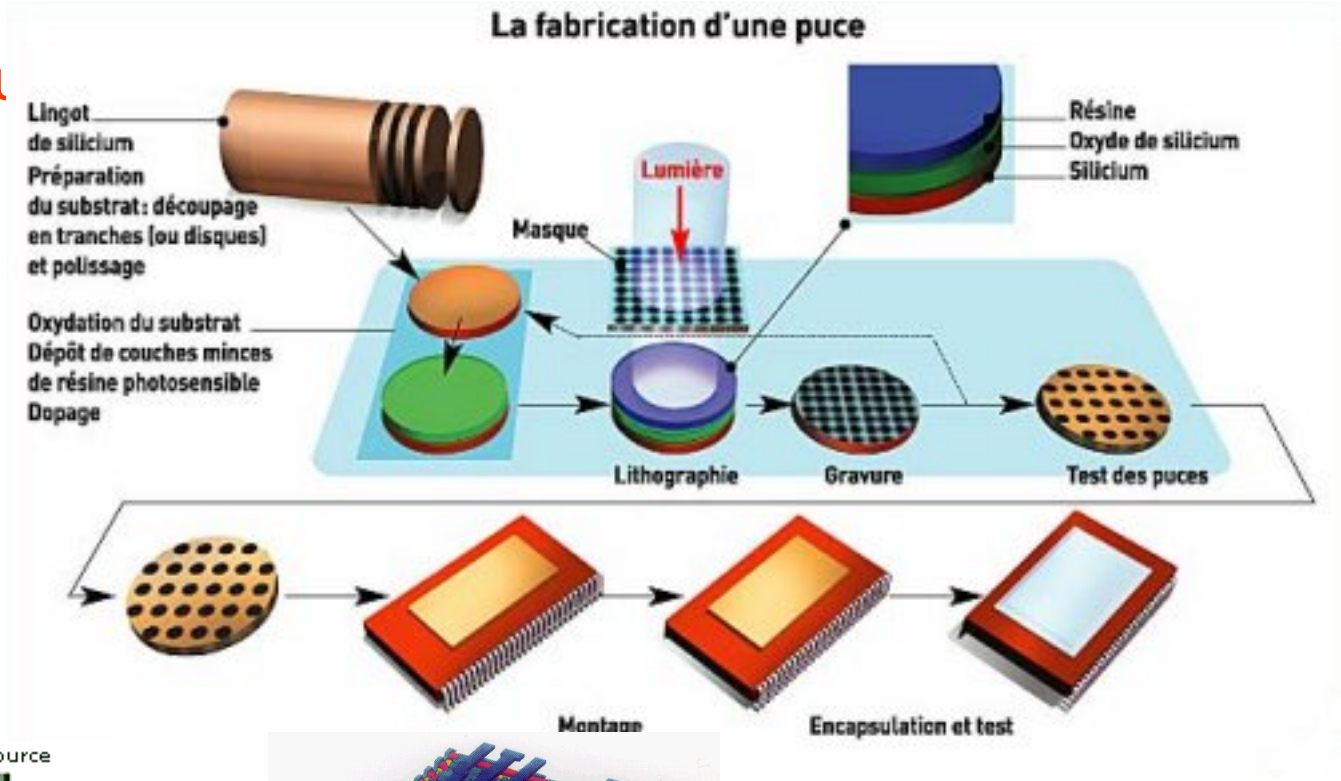
- 1959, 1960 : réalisation par J. KILBY (Texas Instrument) du 1er circuit intégré (consiste à fabriquer dans un même bloc de silicium (une puce) plusieurs composants (résistances, condensateurs, transistors)

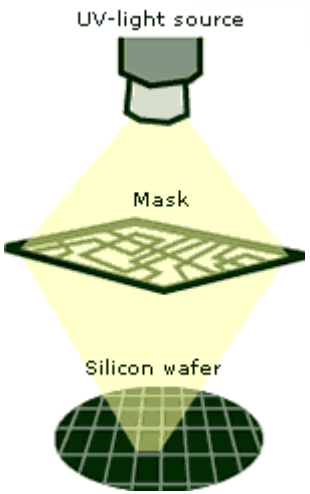


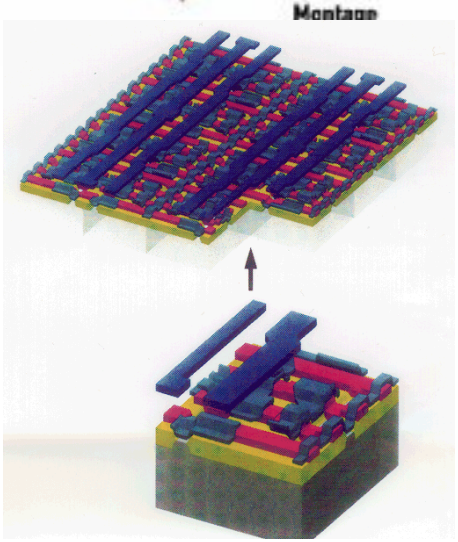
- circuit intégré

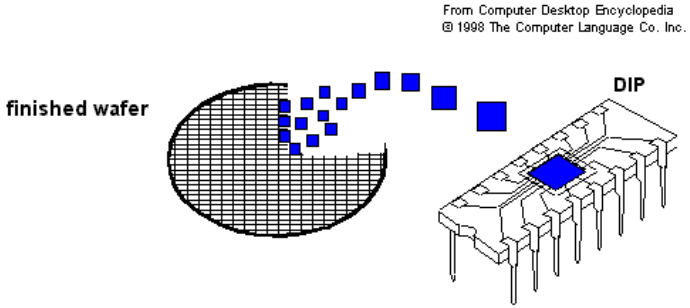


- circu

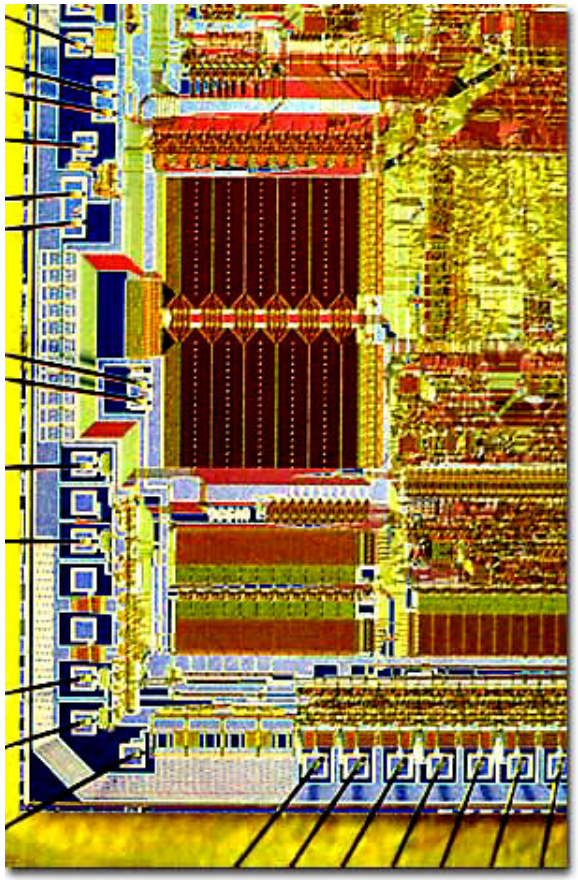

La fabrication d'une puce
Lingot de silicium
Préparation du substrat : découpage en tranches (ou disques) et polissage
Oxydation du substrat
Dépôt de couches minces de résine photosensible
Dopage
Masque
Lumière
Résine
Oxyde de silicium
Silicium
Lithographie
Gravure
Test des puces
Montage
Encapsulation et test

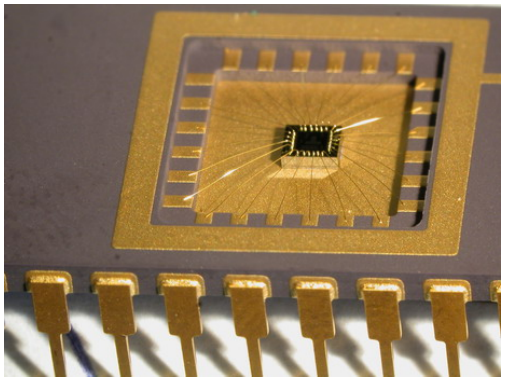

UV-light source
Mask
Silicon wafer




From Computer Desktop Encyclopedia
© 1998 The Computer Language Co. Inc.
finished wafer
DIP

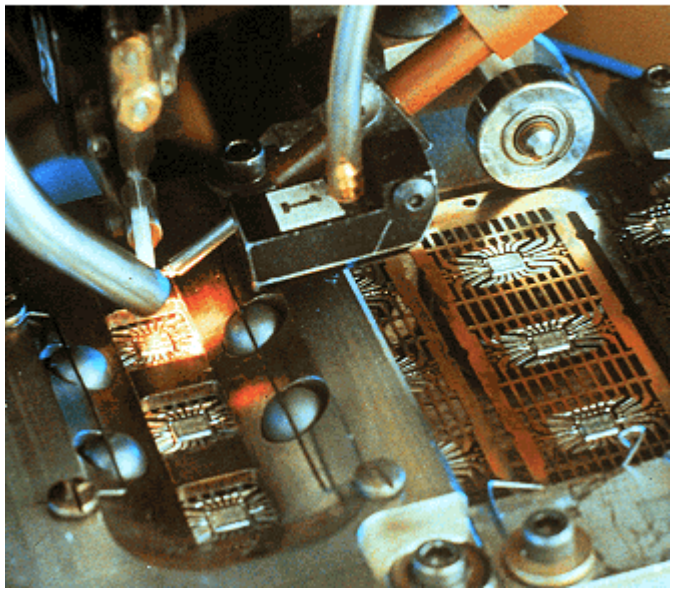
- circuit intégré



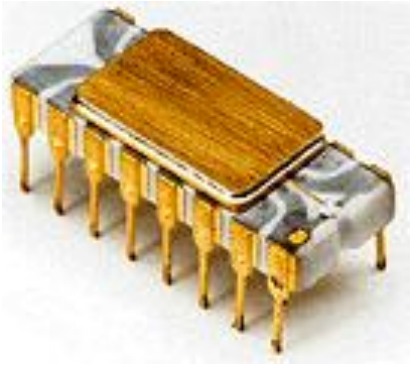


Finesse de gravure
Voir « on line »:
www.hardware.fr

From Computer Desktop Encyclopedia
Reproduced with permission.
© 1996 Texas Instruments, Inc.



- 1971 : le 1er microprocesseur « le 4004 » d'Intel,
  2300 transistors, 4 bits, fréquence de travail 108 kHz (10^3) (à comparer au 3 Ghz (10^9) actuel)



I. Notion de numération binaire (le monde du numérique et de l'analogique)

II. Le modèle Von Neuman et représentation hiérarchique

III. Les différents composants du système et leurs caractéristiques

- **John Von Neumann** (1903 - 1957)
  - mathématicien américain d'origine hongroise.
  - contributions : mécanique quantique, analyse fonctionnelle, en théorie des ensembles, en informatique, en sciences économiques

- Nous utilisons des ordinateurs pour :
  - **Réaliser calculs**
  - **Surveiller et optimiser** un process
  - **Recolorer une image (= faire des calculs)….etc….**

- **exécuter** des programmes = en fait à répéter les actions suivantes :

- Chercher (**FETCH**)
- Exécuter (**EXECUTE**)

- Ces actions sous entendent l'existence de 2 blocs (↓)

- **Fetch and execute**

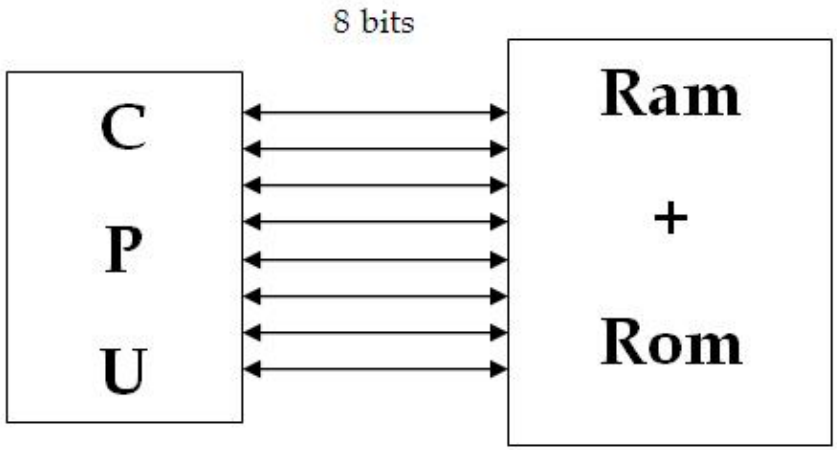


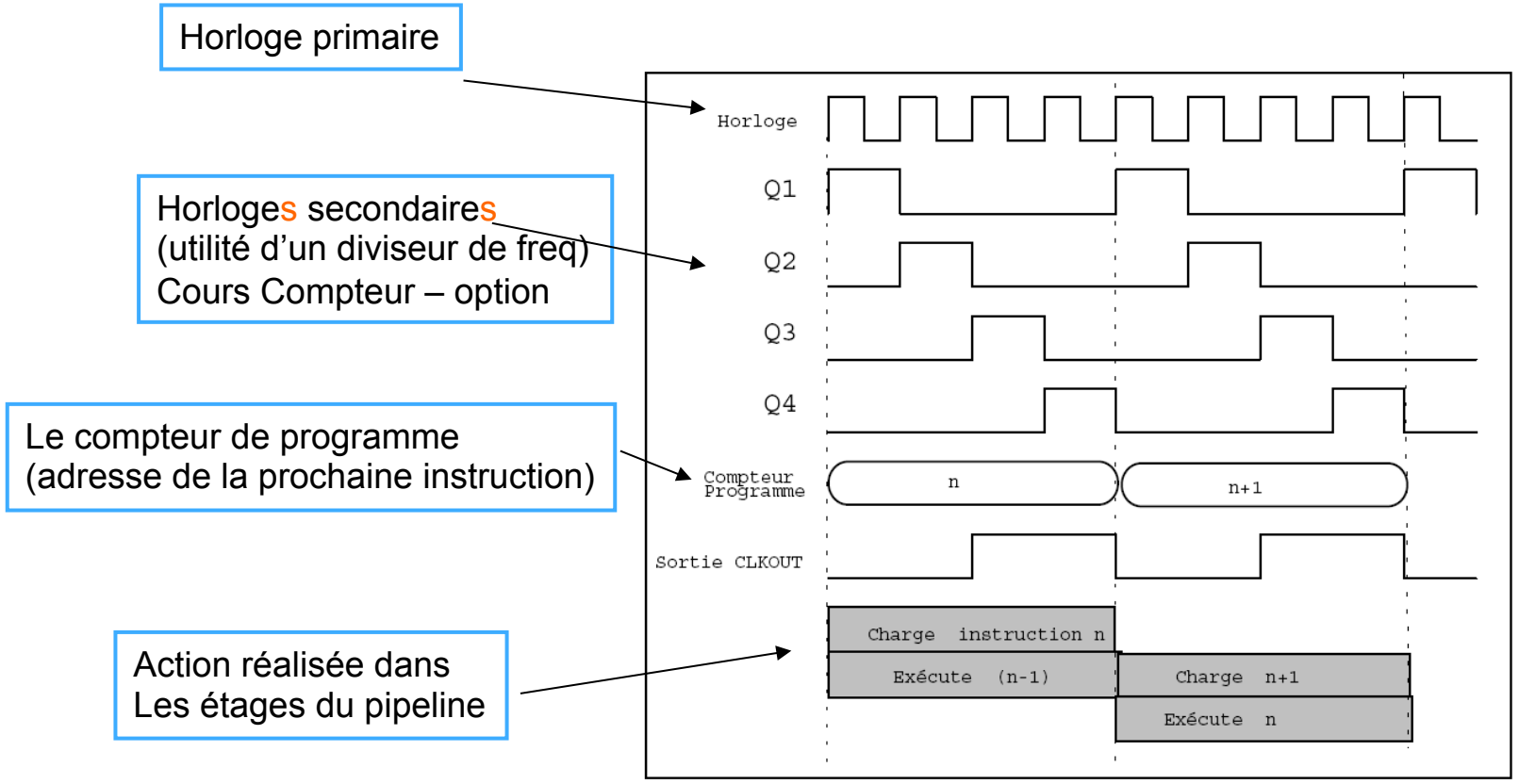
Figure1.1.1

- **Mots à comprendre et retenir :**
  - **Fetch : go for and then bring back (someone or something) : « he ran to fetch help »**
  - **Execute**
  - **Clock**
- **Question : où devez vous stocker vos instructions de programme dans le uP ?**

3.1 Clocking Scheme/Instruction Cycle

The clock is internally divided by four to generate four non-overlapping quadrature clocks, namely Q1, Q2, Q3 and Q4. Internally, the PC is incremented every Q1 and the instruction is fetched from program memory and latched into the instruction register in Q4. It is decoded and executed during the following Q1 through Q4. The clocks and instruction execution flow is shown in Figure 3-3 and Example 3-1.

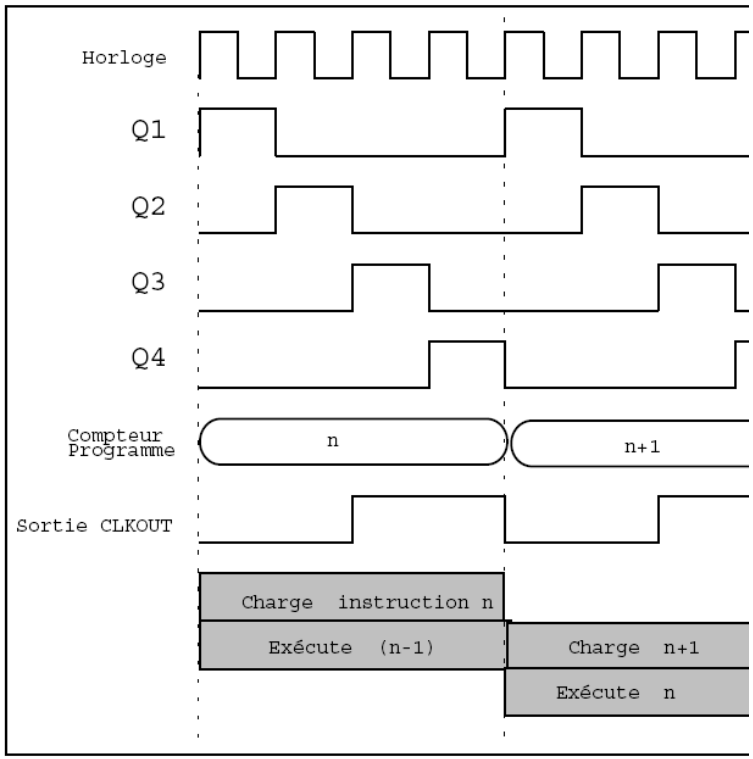
- **Fetch and execute** : Ces actions sont cadencées par l'horloge ou LES horloges.
- Ex PIC fetch execute ⇓



- **Question : où partent les instructions une fois prélever de la mémoire de programme ?**

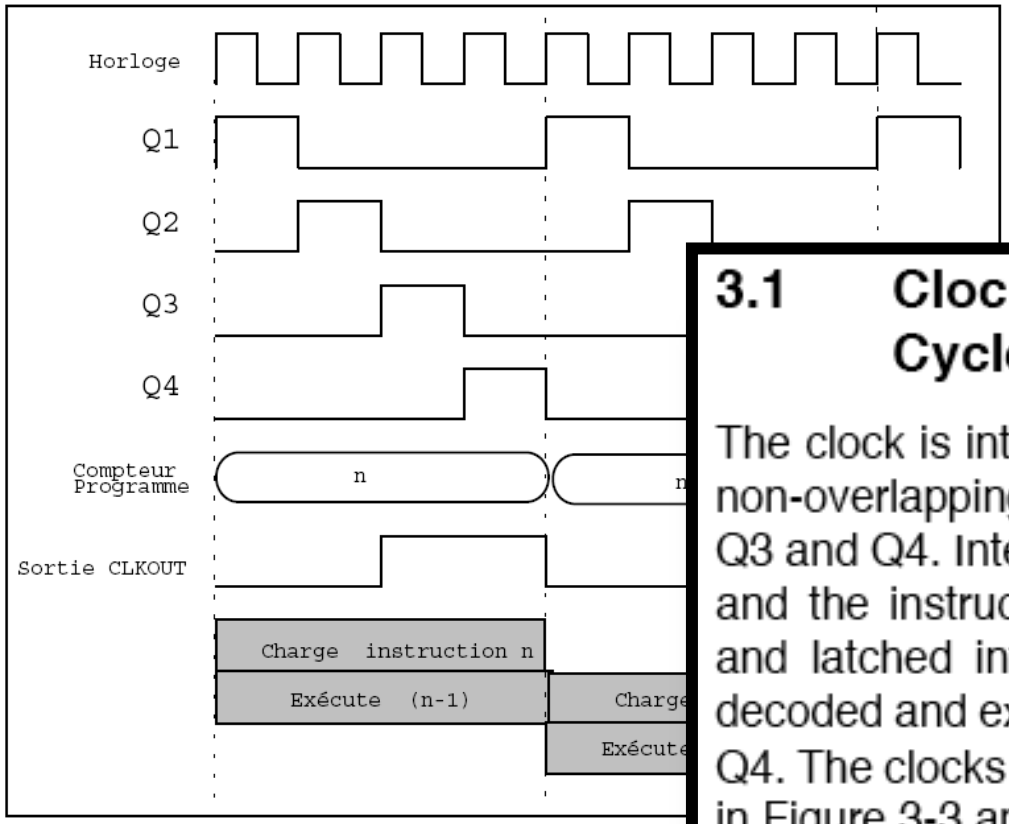
3.1 Clocking Scheme/Instruction Cycle

The clock is internally divided by four to generate four non-overlapping quadrature clocks, namely Q1, Q2, Q3 and Q4. Internally, the PC is incremented every Q1 and the instruction is fetched from program memory and latched into the instruction register in Q4. It is decoded and executed during the following Q1 through Q4. The clocks and instruction execution flow is shown in Figure 3-3 and Example 3-1.



- **Latch =**

- **Question : Le PC Program Counter est un registre**

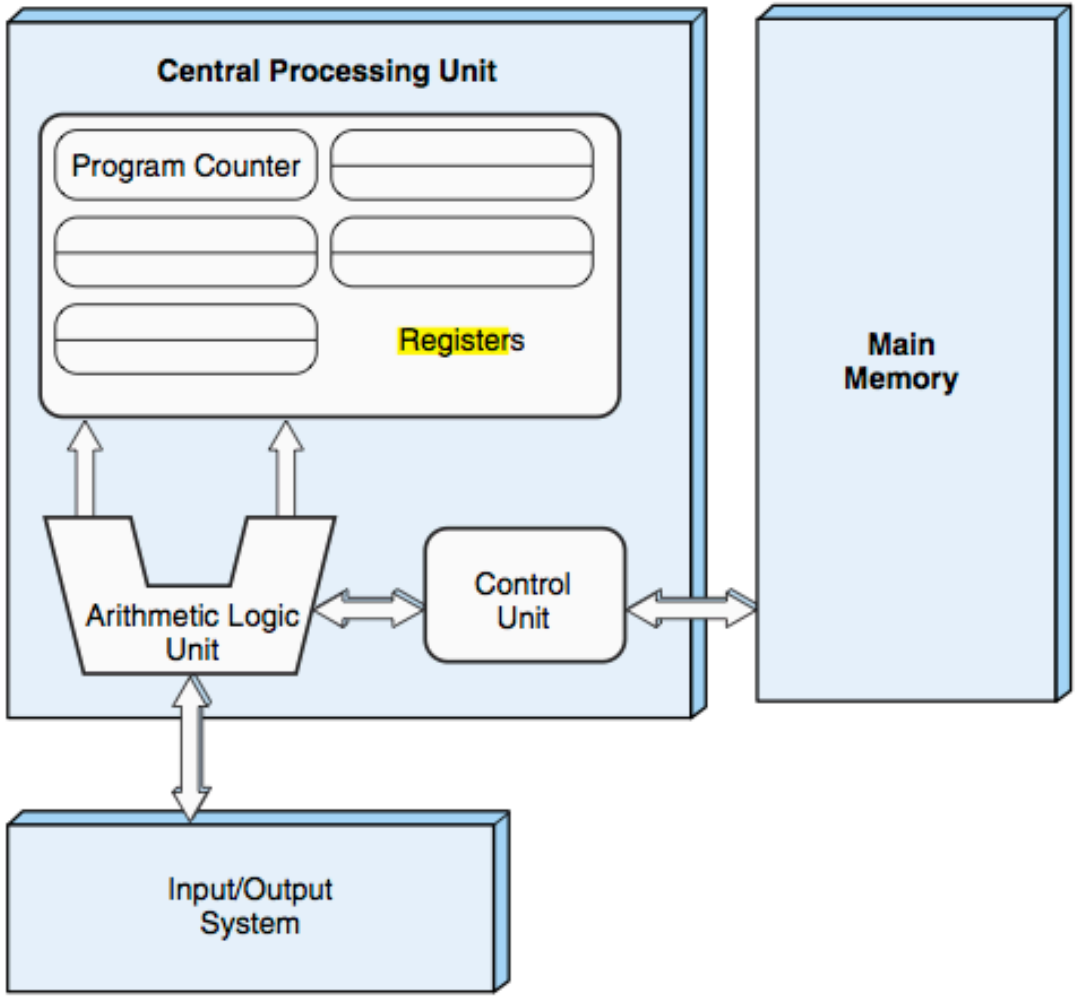


3.1 Clocking Scheme/Instruction Cycle

The clock is internally divided by four to generate four non-overlapping quadrature clocks, namely Q1, Q2, Q3 and Q4. Internally, the PC is incremented every Q1 and the instruction is fetched from program memory and latched into the instruction register in Q4. It is decoded and executed during the following Q1 through Q4. The clocks and instruction execution flow is shown in Figure 3-3 and Example 3-1.

Terme a retenir PC

- **Question : Le PC Program Counter - localisation**

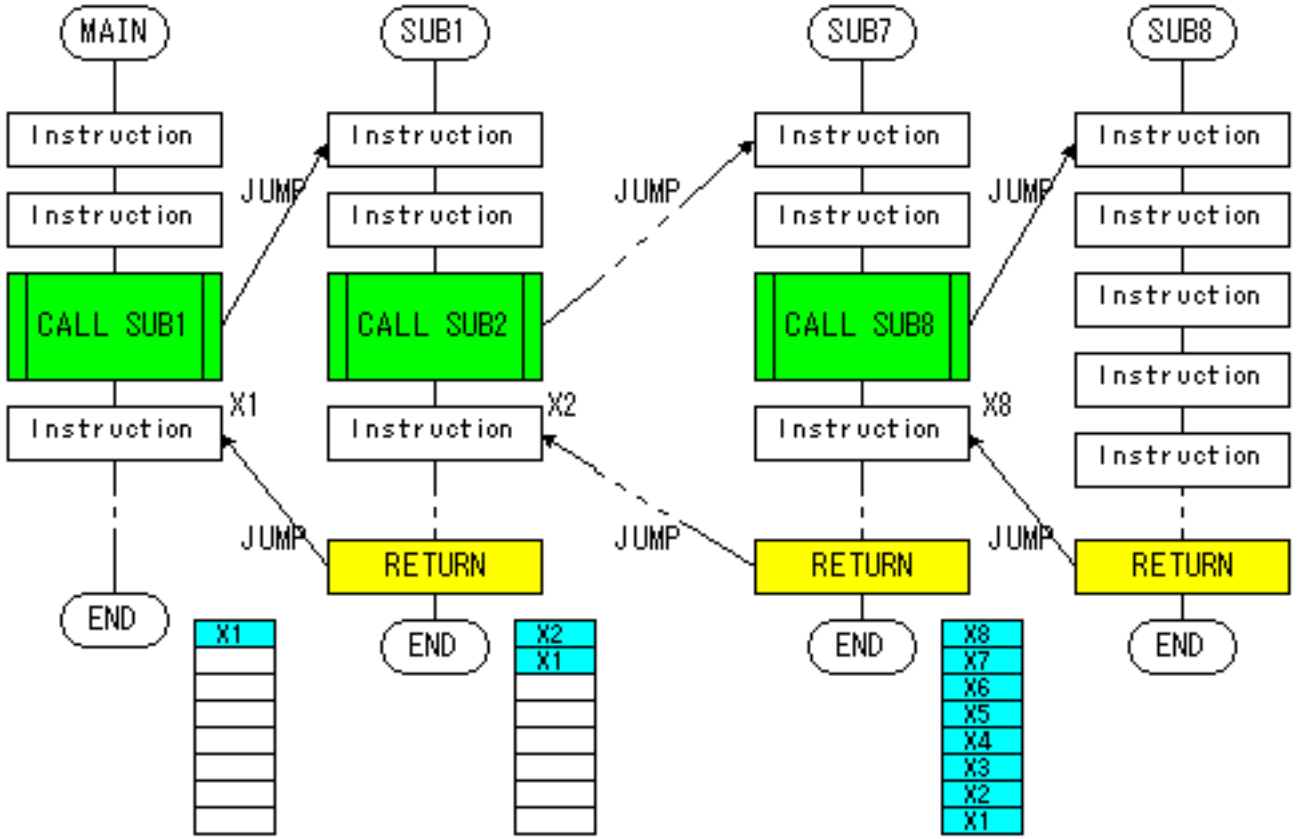


- **Question qu'est-ce qu'un registre ?**

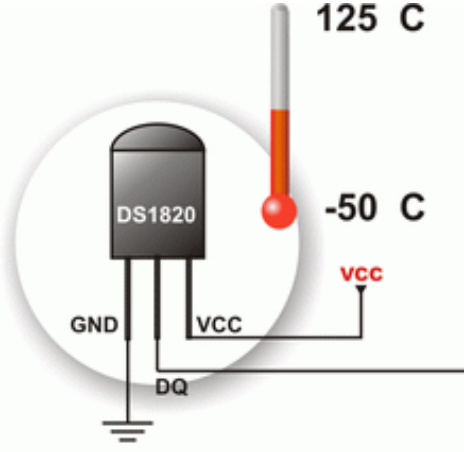
The Registers

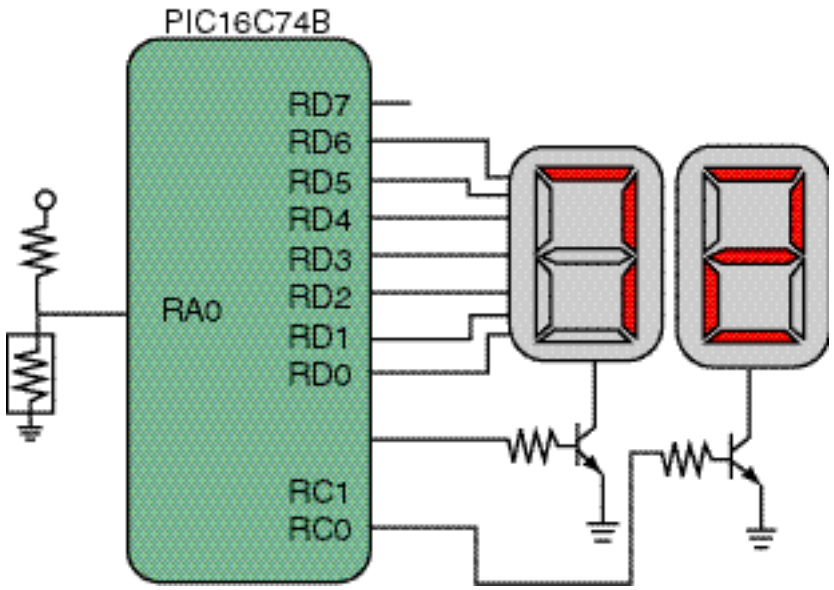
Registers are used in computer systems as places to store a wide variety of data, such as addresses, program counters, or data necessary for program execution. Put simply, a *register* is a hardware device that stores binary data. Registers are located on the processor so information can be accessed very quickly. We saw in Chapter 3 that D flip-flops can be used to implement registers. One D flip-flop is equivalent to a 1-bit register, so a collection of D flip-flops is necessary to store multi-bit values. For example, to build a 16-bit register, we need to connect 16 D flip-flops together. We saw in our binary counter figure from Chapter 3 that these collections of flip-flops must be clocked to work in unison. At each pulse of the clock, input enters the register and cannot be changed (and thus is stored) until the clock pulses again.

- **LE PC** n'est **pas incrémenter à +1** quand il y a un appel à sous programme.
- Utilité d'un **sous programme** ?



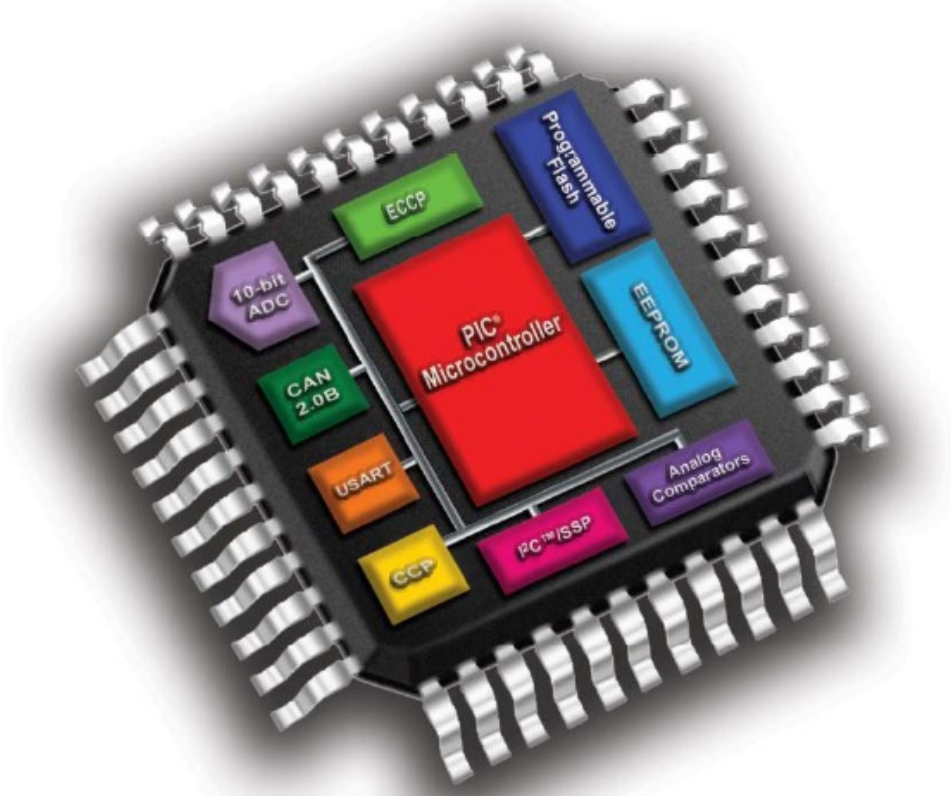
- Exemple de réalisations nécessitant un sous programme « tempo » faisant « sauter » le PC
- CdC : nous voulons réaliser un système qui **relève la température toutes les 5s**
- Prog « tempo » = programme dont l'exécution va prendre 5s

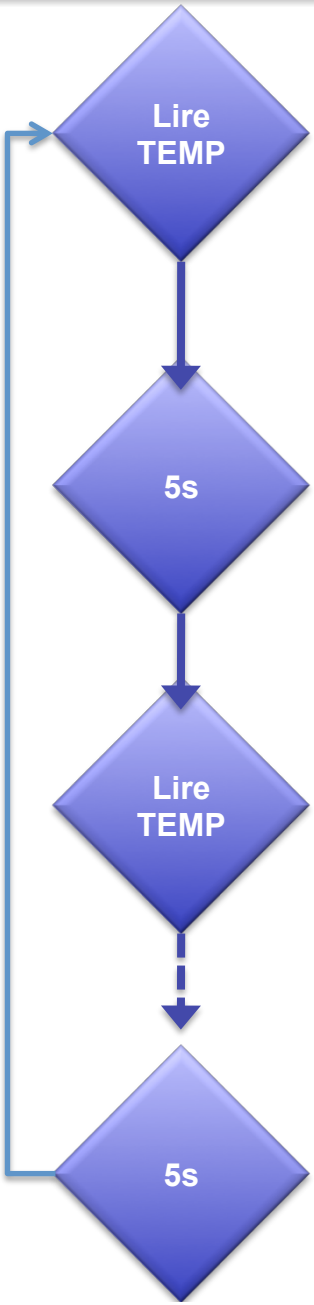




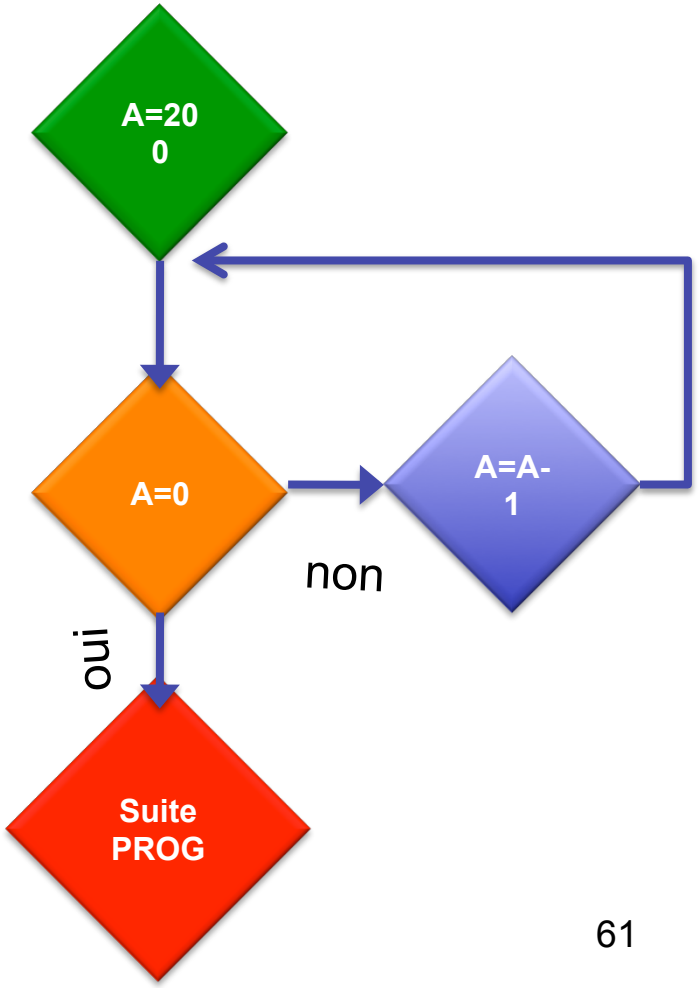
- Déroulement/organigramme:

- Question : Lire TEMP peut, elle aussi être une sous fonction (= comment lit-on une température) ? (CAN 10F200)





- Comment faire Prog « tempo » ? = programme dont l'exécution va prendre 5s … ne fait rien d'autre.
- Algorithme associé :
  - Stocker 200 dans le registre A
  - Tester si A=0 ?
    - Si non faire A=A-1 et tester A=0 ?
    - Si oui passer à la suite du programme



- Est basé sur le fait qu'exécuter une instruction prends X(ms)
- Exemple PIC(↓)

- Prog « tempo » = programme dont l'exécution va prendre 5s
- Instructions clés : decfsz et goto

```
;Delay of 5ms approx. Instruction cycle time is 5us.
delay5  movlw       D'200'          ;200 cycles called,each taking 5x5=25us
        movwf       delcntr1
del1    nop                         ;1 inst. cycle
        nop                         ;1 inst. cycle
        decfsz      delcntr1,1      ;1 inst. cycle, when no skip
        goto del1                   ;2 inst. cycles
        return
```

Program Example 5.2 A delay subroutine

DECF	**Decrement f**
Syntax:	[*label*] DECF f,d
Operands:	$0 \le f \le 31$ $d \in [0,1]$
Operation:	(f) – 1 → (dest)
Status Affected:	Z
Description:	Decrement register 'f'. If 'd' is '0', the result is stored in the W register. If 'd' is '1', the result is stored back in register 'f'.

1ère aperçu de l'assembleur

- L'architecture Von Neumann
  - Modélise un mode de fonctionnement et la structure d'un ordinateur
  - utilise une structure de stockage **unique** pour conserver à la fois :
    - les **instructions**
    - et les **données** requises

    pour réaliser par le calcul. Ces instructions ou données sont stockées dans des **cases mémoire** repérées par une **adresse binaire**.
  - Se caractérise aussi par la séparation entre le stockage et le processeur.

  - Plus tard **architecture Harvard** : **séparation des zones** de **données** et de **programmes**
    - Avantages ???
    - Inconvénients ??

- L'architecture Von Neumann
  - **zones** de **données** et de **programmes** sont dans même « composant/ partie »

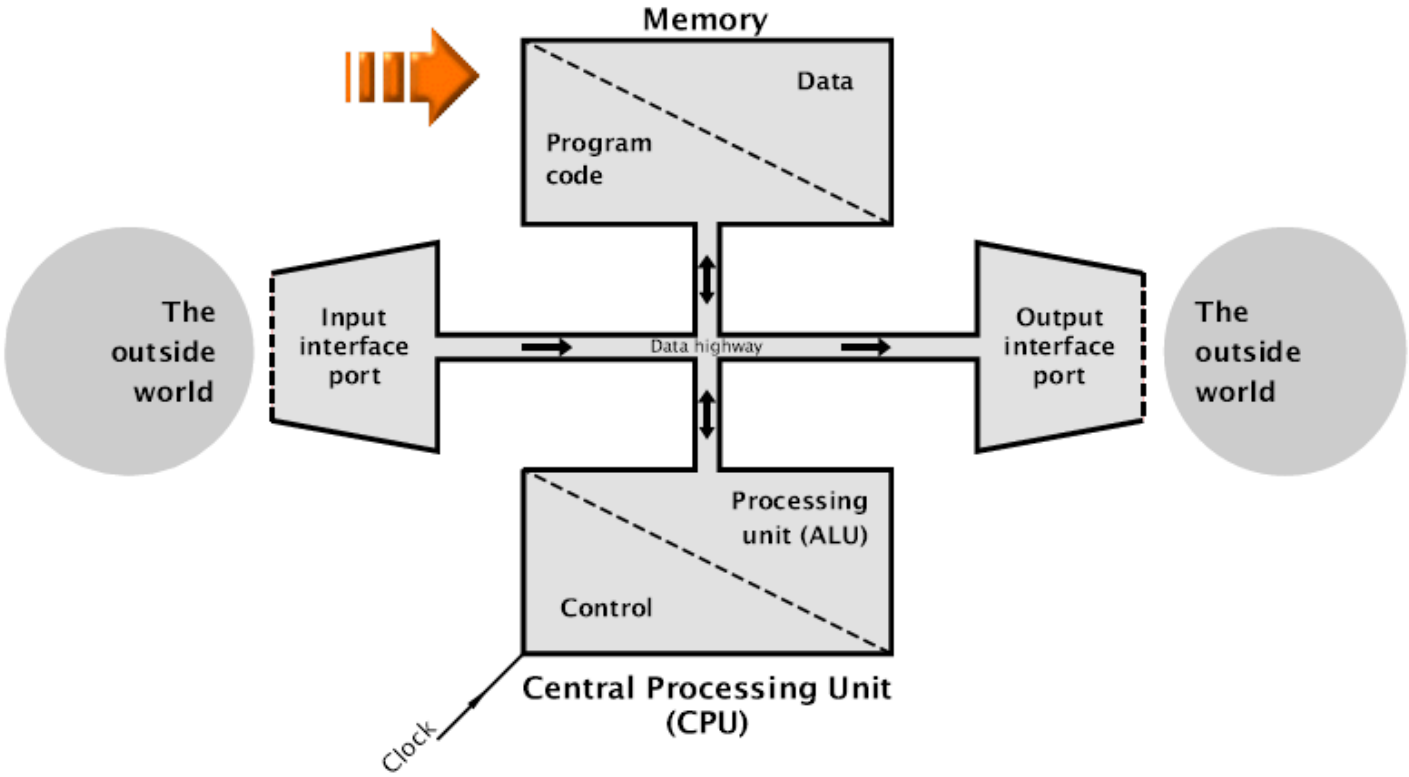


Fig. 3.1 An elementary von Neumann computer.

- L'architecture Harvard
  - **zones** de **données** et de **programmes : séparées**[1]
  - **Bus de données, programme : séparés** (**ex :** architecture PIC(🡻))

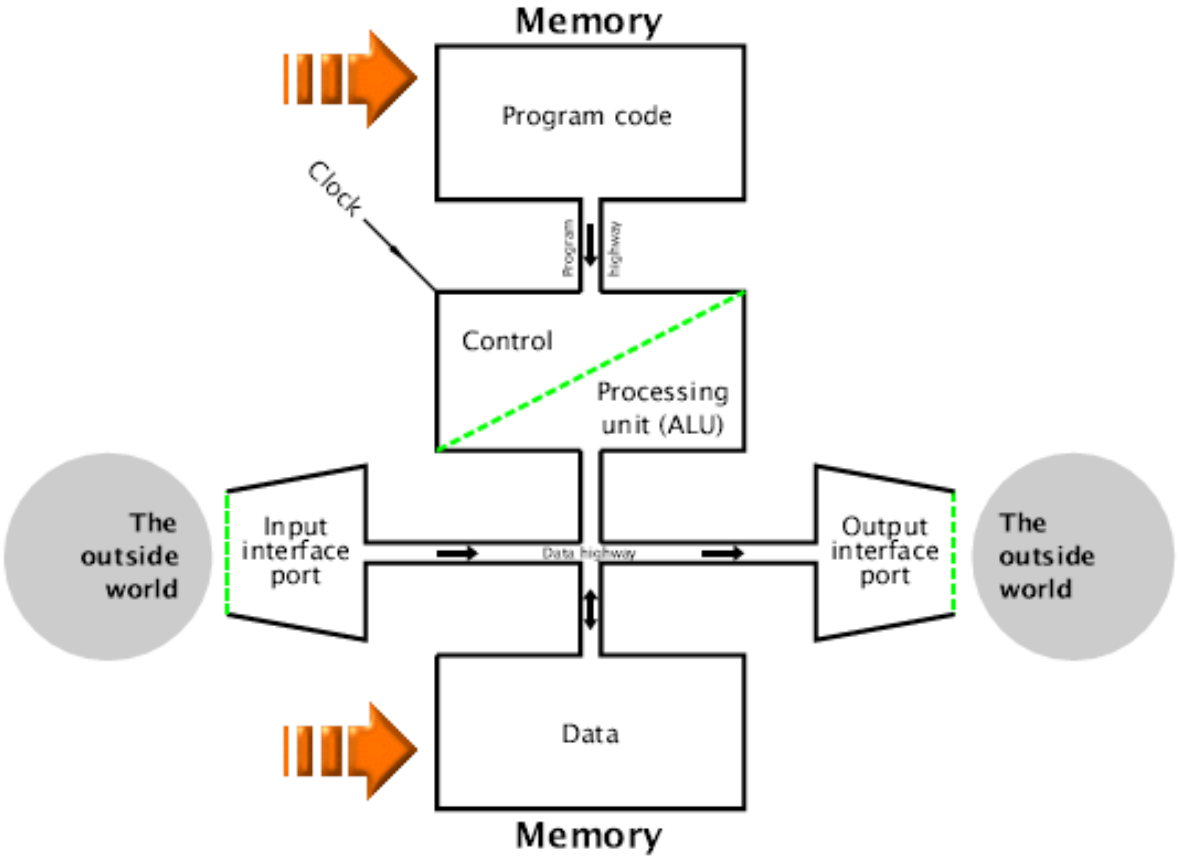
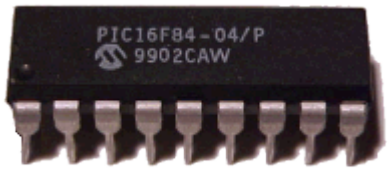


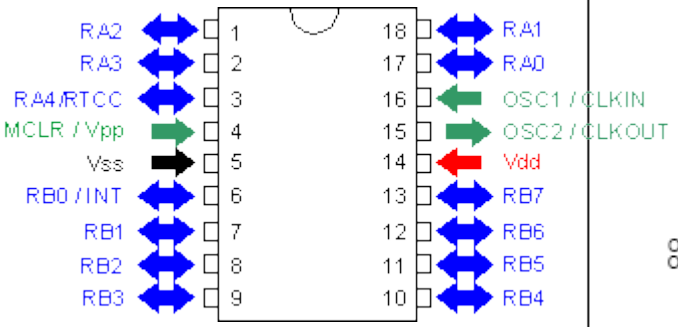
Fig. 3.2 An elementary Harvard architecture computer.

[1]*Quintessential pic microcontroller*

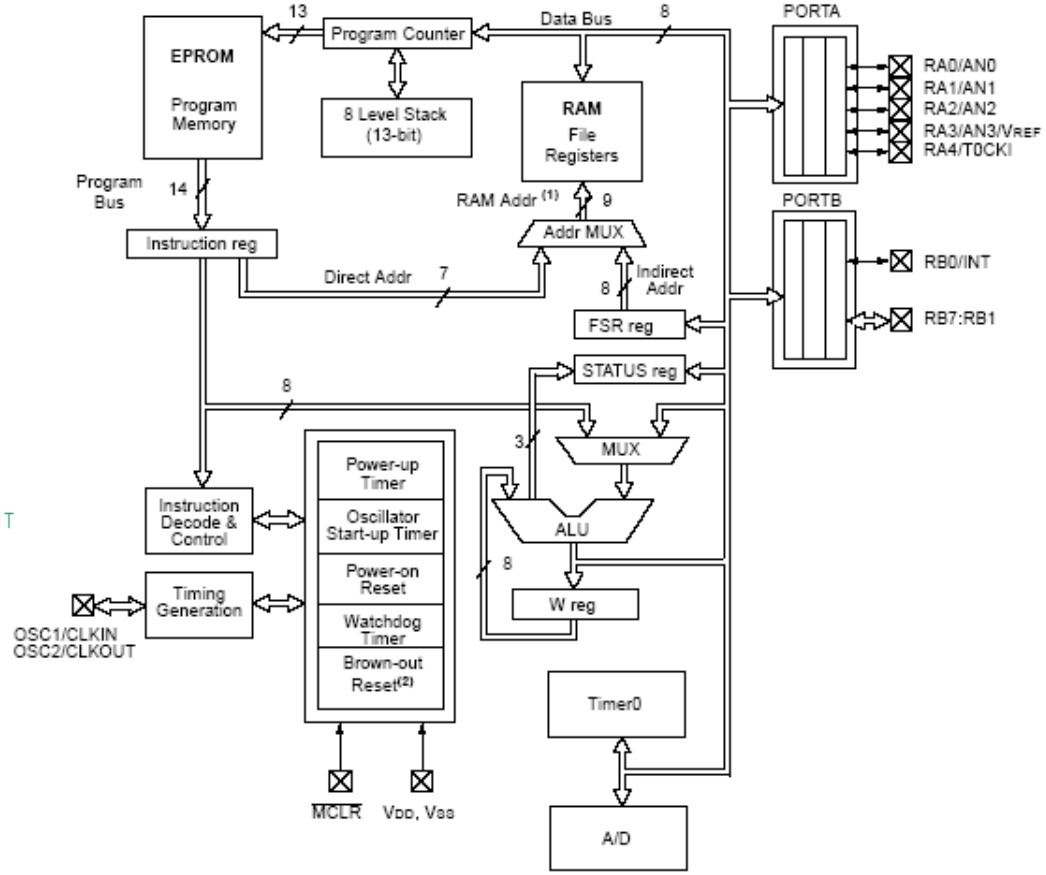
- Première vision du PIC en terme d'architecture (ZOOM IN🡻)



localiser mémoire de programme et de données

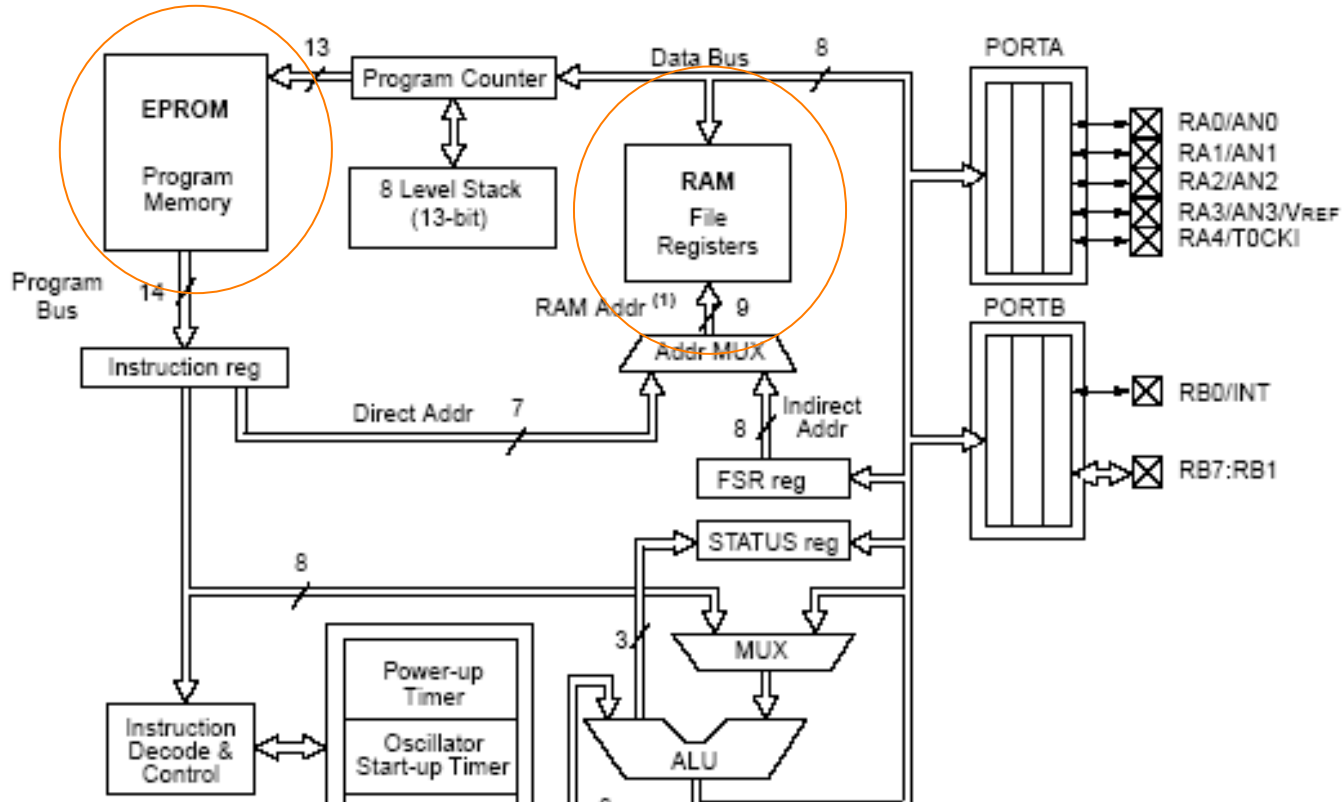


Device	Program Memory	Data Memory (RAM)
PIC16C710	512 x 14	36 x 8
PIC16C71	1K x 14	36 x 8
PIC16C711	1K x 14	68 x 8
PIC16C715	2K x 14	128 x 8

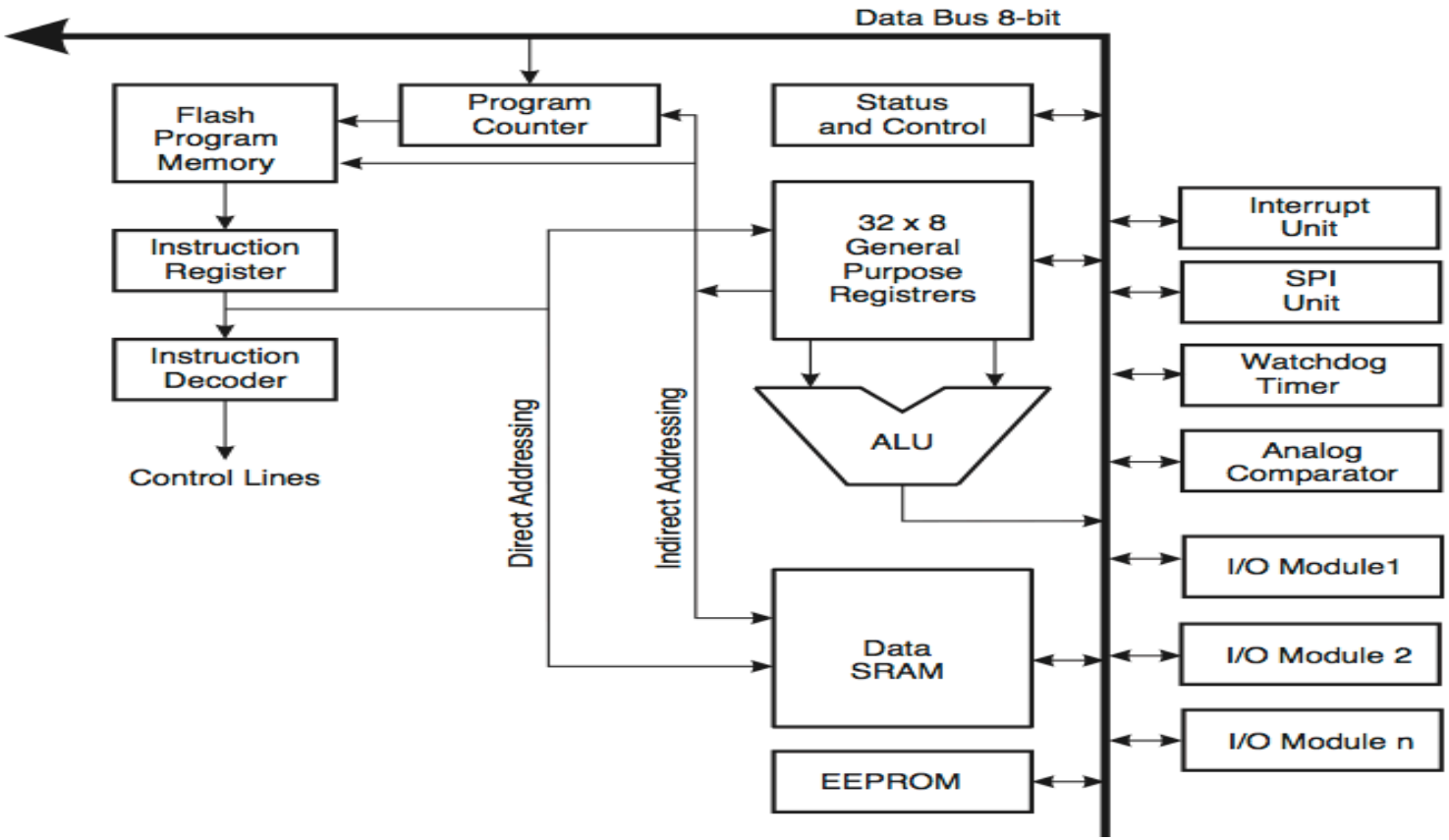


- (ZOOM IN🡻) : zone mémoire de programme et de données
- ⇒ les PICs ont **structure Harvard**

Device	Program Memory	Data Memory (RAM)
PIC16C710	512 x 14	36 x 8
PIC16C71	1K x 14	36 x 8
PIC16C711	1K x 14	68 x 8
PIC16C715	2K x 14	128 x 8

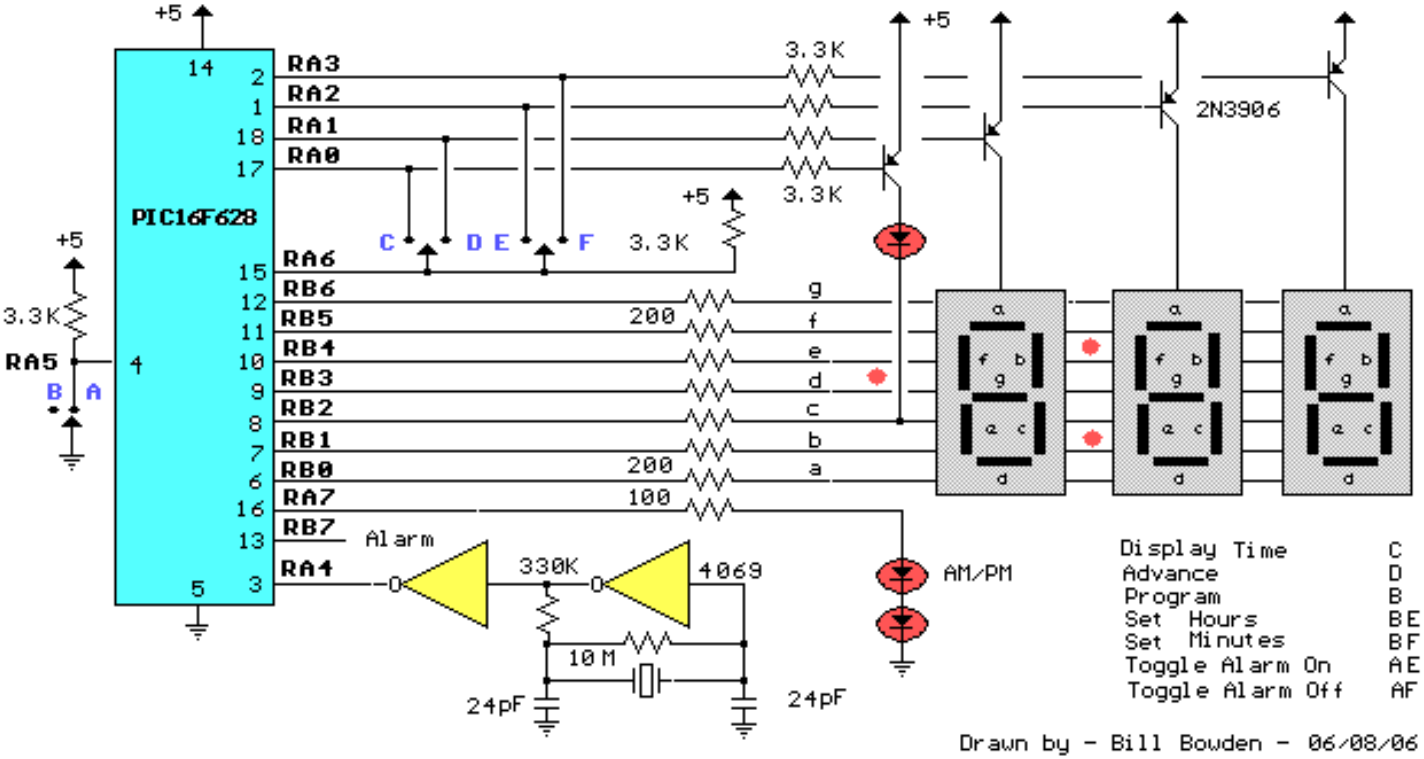


- Cas de la carte UNO et de son microcontroleur ATMEGA 328



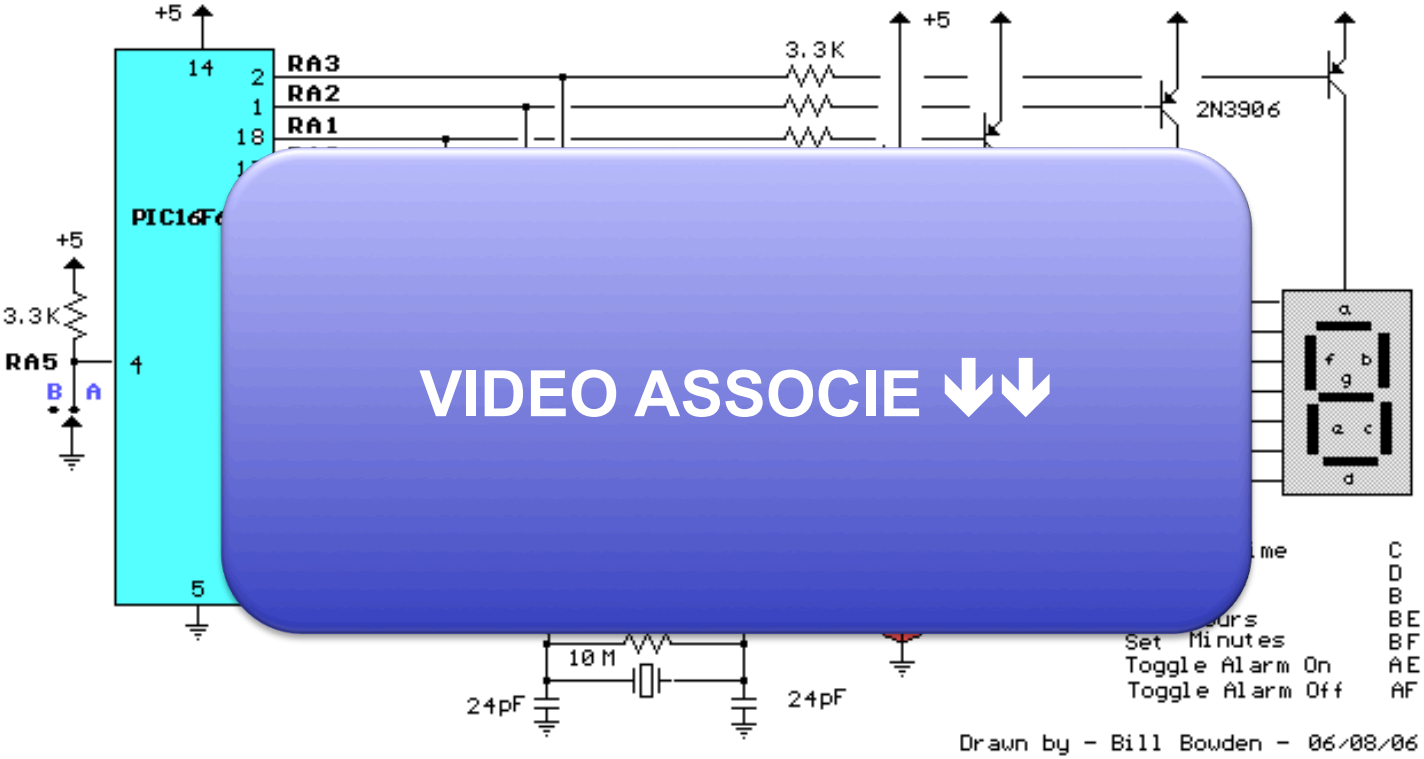
In order to maximize performance and parallelism, the AVR uses a Harvard architecture – with separate memories and buses for program and data. Instructions in the program memory are executed with a single level pipelining. While one instruction is being executed, the next instruction is pre-fetched from the program memory. This concept enables instructions to be executed in every clock cycle. The program memory is In-System Reprogrammable Flash memory.

- **ETUDE DE CAS 1-CM2 :**
  - **Ports** numériques I/O d'un uC : **RBi** utilisés pour le pilotage **d'afficheur 7 segments** par exemple



  - **PORTB = ensemble des Rbi**
  - **Comment compter de 000 à 999 avec cette connectique ?**

- ETUDE DE CAS 1-CM2 :
  - **Ports** numériques : **RBi** utilisés pour le pilotage **d'afficheur 7 segments** par exemple



- **PORTB = ensemble des Rbi**
- **Comment compter de 000 à 999 avec cette connectique ?**

- ETUDE DE CAS 1 –CM2 :
  - **Ports** numériques : **RBi** utilisés pour le pilotage **d'afficheur 7 segments** par exemple : video « video_2 digit 7 segment multiplexing principle » .. Persistance rétinienne.

2 digit 7 segment multiplexing principle

  - Sur le même principe video « video_LED Cube 8x8x8 running on an Arduino.mp4 … comment gérer les commandes pour minimiser le nombre de fils de connexion ???