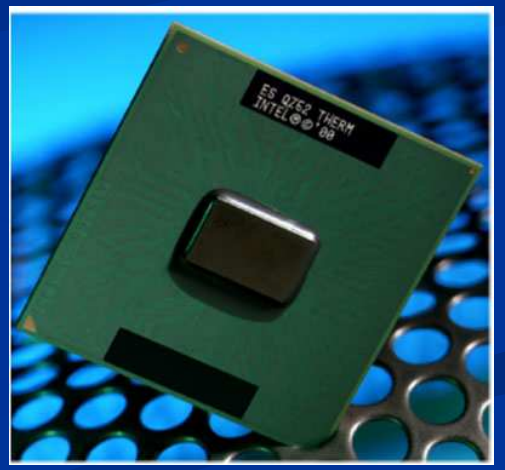
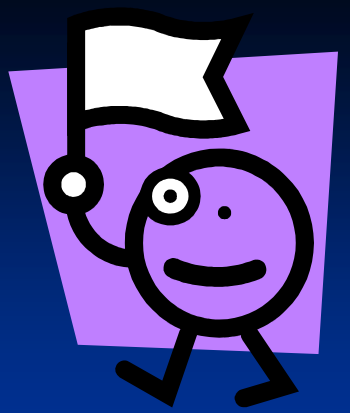
Microcontrôleurs

Le monde du BIT de la microseconde et du nanomètre



Christian Dupaty
Académie d'Aix-Marseille
christian.dupaty@ac-aix-marseille.fr

Sommaire



- SAM1A :
  - Introduction
  - Numération binaire
  - Systèmes microprogrammés
  - Technologie des microcontrôleurs
  - Les outils de développement
  - Notion d’algorithmie, programmation structurée
  - Programmation en assembleur
  - Périphériques parallèles, séries, TIMER, UART
  - Mise en œuvre des interruptions
  - Projet technique en binôme

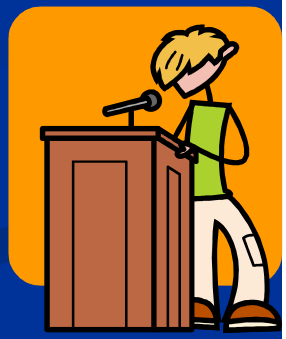
Objectifs

- Être capable de concevoir un programme simple en langage assembleur pour microcontrôleur INTEL 8051
- Analyse, algorigrammes, programmation structurée, jeux d'instructions, modes d'adressage
- Mise en œuvre de l'outil de développement KEIL pour 8051 et du simulateur PROTEUS/ISIS

Evaluation

- Test écrit individuel de 1h30 . (coef 3)
  - Connaissances théoriques
  - Rédaction d'un programme en assembleur

- Soutenance orale et individuel du dossier du projet technique réalisé en binône. (coef 1)
  - Autonomie, implication, vérification de la connaissance du dossier.



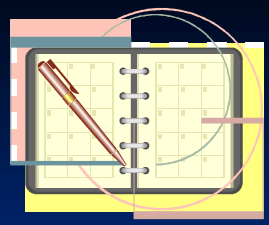
Conditions d'apprentissage

- 8 séances de cours / TP (24h)
- Mise à disposition des logiciels de développement KEIL uVISION2 et ISIS/PROTEUS pour le travail personnel.
- Ponctualité et assiduité indispensable
- En cas d'absence non justifiée pas d'examen de rattrapage.

Ressources

- Cours SAM1A EMSE
- WWW.infineon.com Products - Microcontrollers - C500 - C517A
- WWW.keil.com Evaluation software - C51 (8051) – Tools
- WWW.phytec.de English version – Products – Microcontrollerboards - Kitcon C517A
- « Microcontrôleurs 8051 et 8052 » de Bernard Odant chez DUNOD
- « Les microcontrôleurs SAB 80C515 / 80C535 et leurs programmation » de H. Ezzedine & M. Abed chez MASSON
- Mise en œuvre et application du microcontrôleur 8051 « de P. Kauffman chez MASSON

Planing



- S1 : Cours pages chap1-chap10, début TP1
- S2 : TP1 outil de développement, simulateur, exercices gestion, mémoire, ports parallèles
- S3 : TP2 utilisation avancée du simulateur, gestion d'une table, opérations +-*/, présentation de VSM (ISIS, uVision2)
- S4 : Cours pages chap 11, TP3 : interruptions (INT0, 1)
- S5 : Cours TIMER 0,1 chap 12, TP4 : production de signaux, comptage, PWM
- S6 : Cours TIMER 2 chap 15, fonction capture TP5 : mesure de durées, de périodes
- S7 : Cours communications, TP6 sur simulateur
- S8 : Bilan, révisions, présentation et début des projets.

1

Microprocesseurs VS Microcontrôleurs

- Microprocesseurs, ex :PENTIUM, AMD, 68000



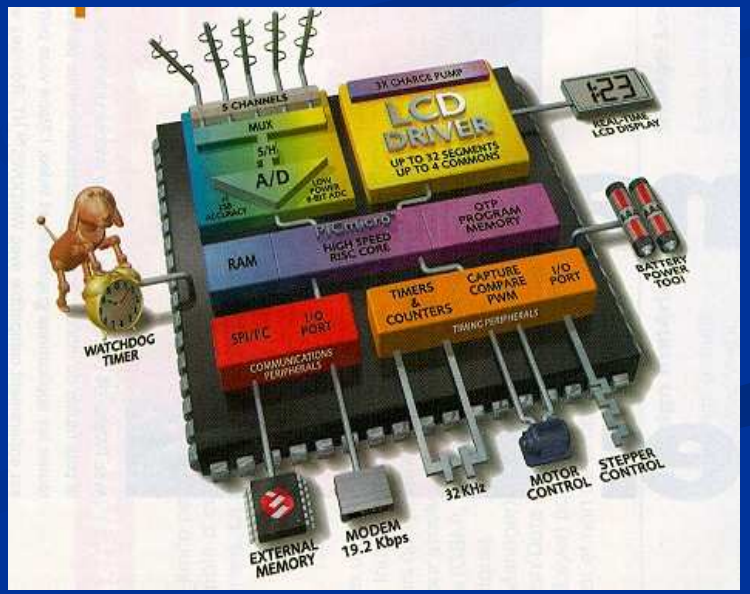
- Microcontrôleurs , ex : 8051, 68HC11, PIC18, AVR, ST6 ...






Microcontrôleurs - Avantages

- Faible cout
- Reconfigurable (modification du programme)
- Fiabilité, tout est sur la même puce (processeur, mémoires, périphériques)
- Faible consommation



Domaines d'emplois: Petits automatismes

Domaines d'emplois: Traitement numérique du signal (Digital Signal Processor)

Acquisition,
conditionnement
Echantillonnage
Conversion analogique-numérique
Codage,
Enregistrement

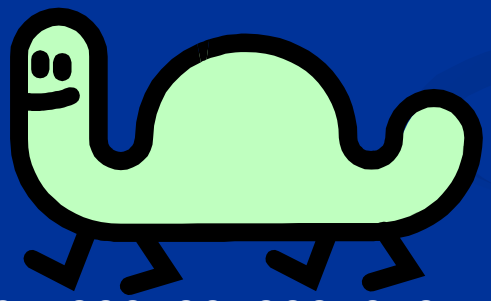
Lecture
Decodage
Conversion Numérique – Analogique
Conditionnement
Amplification

Caractéristiques uP/uC

	Petits automatismes	Informatique industrielle
Fréquences	4 à 40MHz	400MHz à 4.0GHz
Puissance de calcul	faible	Très élevé
Mémoire vive	RAM : 1KO	RAM : 512MO ROM : 1MO
Mémoire programme	ROM FLASH: 64 KO	Disque dur
Intégration des périphériques	Maximum	minimum
consommation	faible	Elevé
prix	2€ à 50€.	200€ à 1000€
Composant utilisé	Microcontrôleur 8/16 bits	Microprocesseur 32/64 bits

Eléments de numération binaire

Les origines des calculateurs



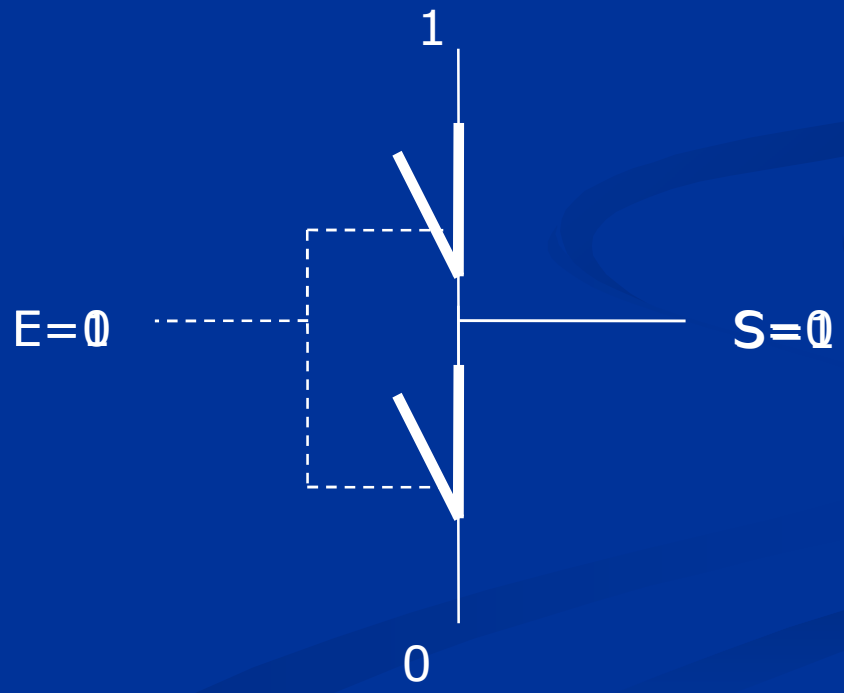
01100100110110111101011000100100010101100010010110001111001 01

Le BIT : 0 ou 1

- En électronique numérique toutes les opérations s'effectue en base 2
- Aux deux états logiques sont associés deux tensions :
  - En logique « positive » 0 ➔ 0v et 1 ➔ 1v à 5v
  - En logique « négative » 1 ➔ 0v et 0 ➔ 1v à 5v

Interrupteur électronique

- Ces deux états électriques dépendent de la conduction ou du blocage d'un composant électronique
- Ex : Fonction Booléenne NON, l'inverseur

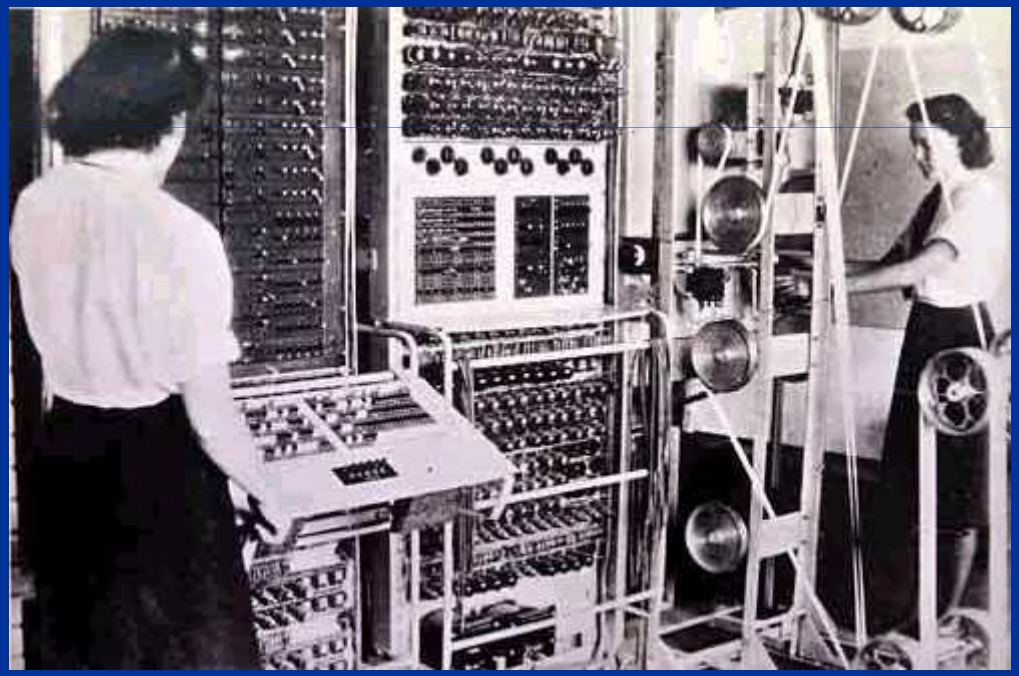


AVANT

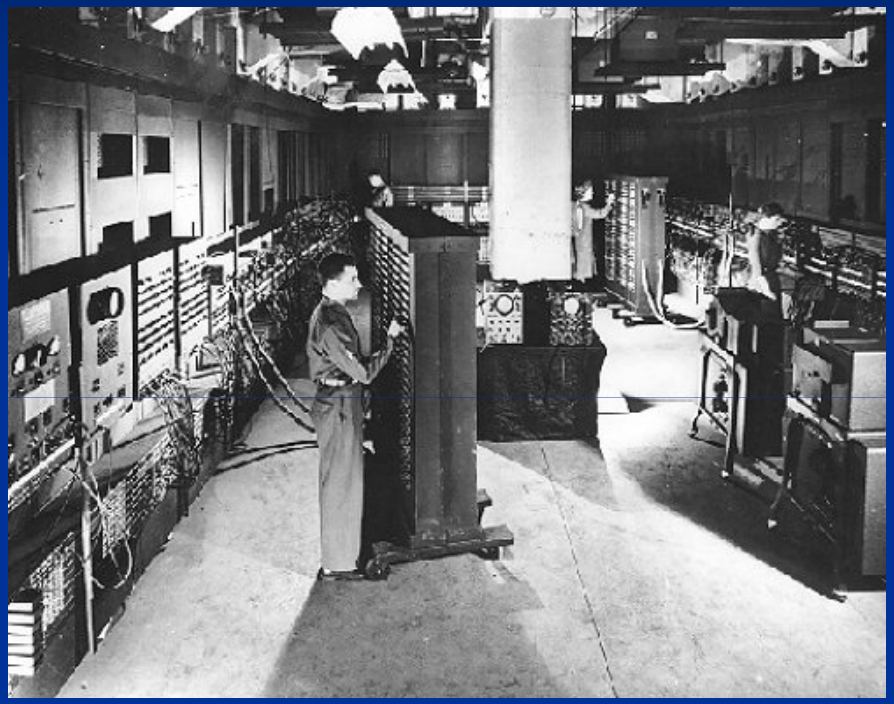
- TUBE TRIODE (Lee De Forest en 1906.)

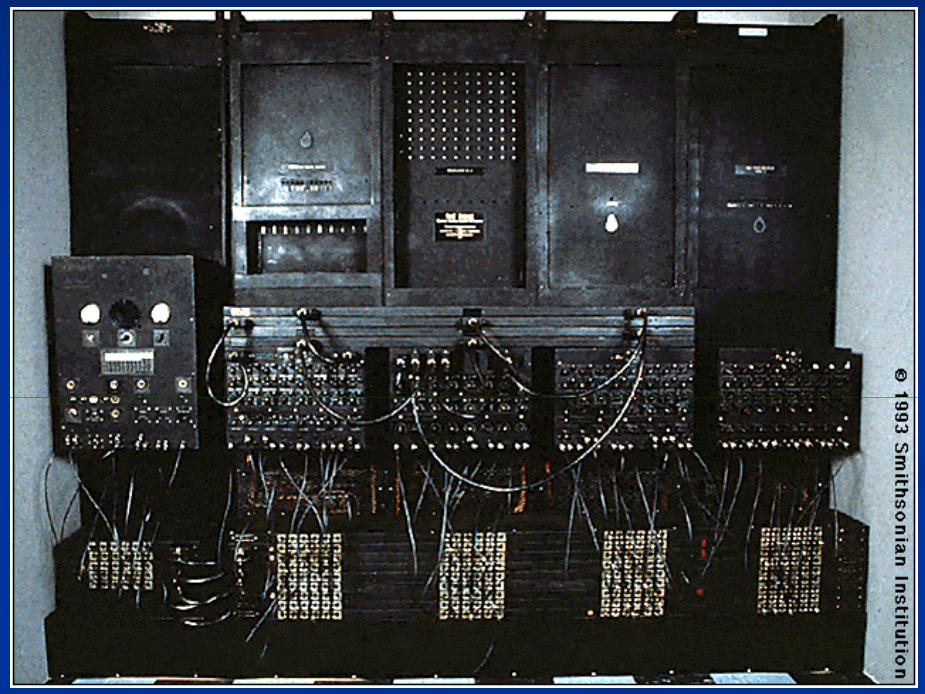
1940 Z3, COLOSSUS

Inventés par Konrad Zuse et Arnold Lynch
Ordinateurs électromécaniques (commutateurs à relais)
fréquence du processeur : 5,33 Hz
puissance réelle : 20 Flops



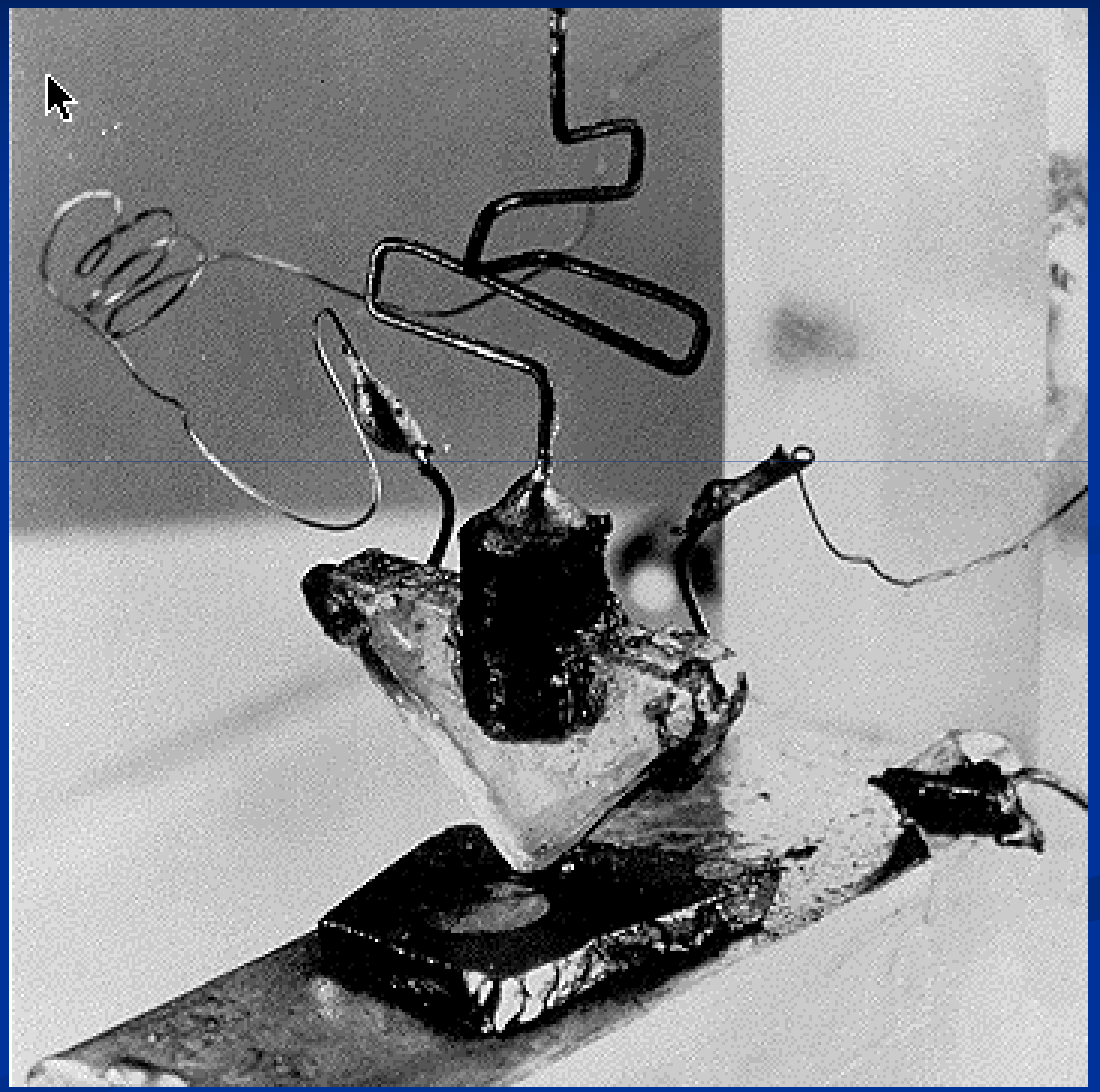
Premier Ordinateur ELECTRONIQUE– ENIAC 1946





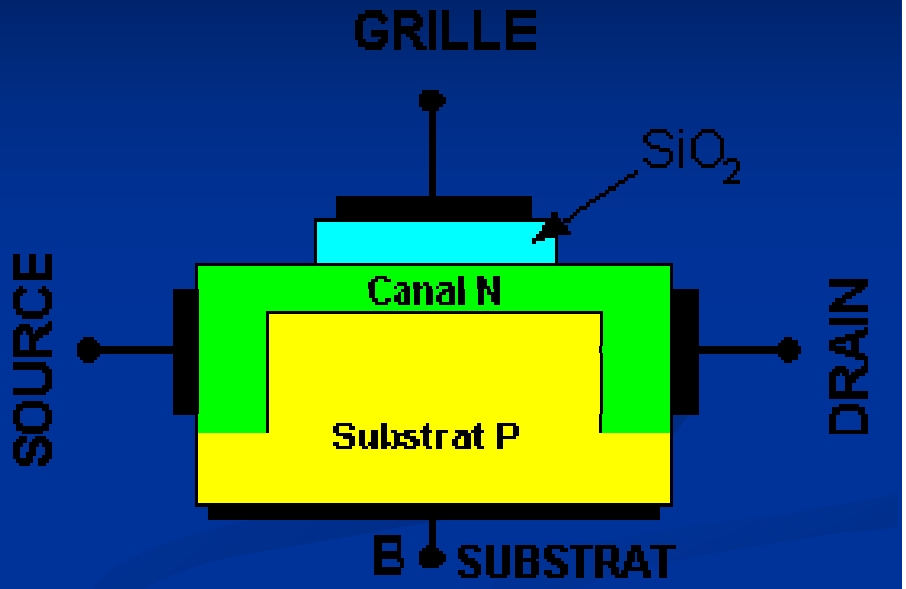
17 468 tubes à vide, 7 200 diodes à cristal, 1 500 relais, 70 000 résistances, 10 000 condensateurs et environ 5 millions de joints soudés à la main. Son poids est de 30 tonnes.
5 000 additions simples ou 357 multiplications ou 38 divisions par seconde.

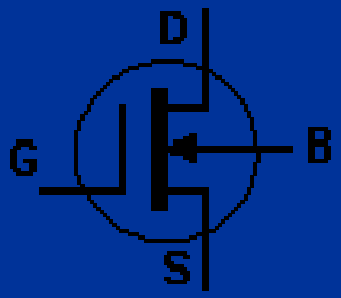
L’invention du Transistor 1947

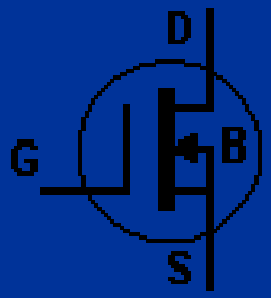


Principes du MOS

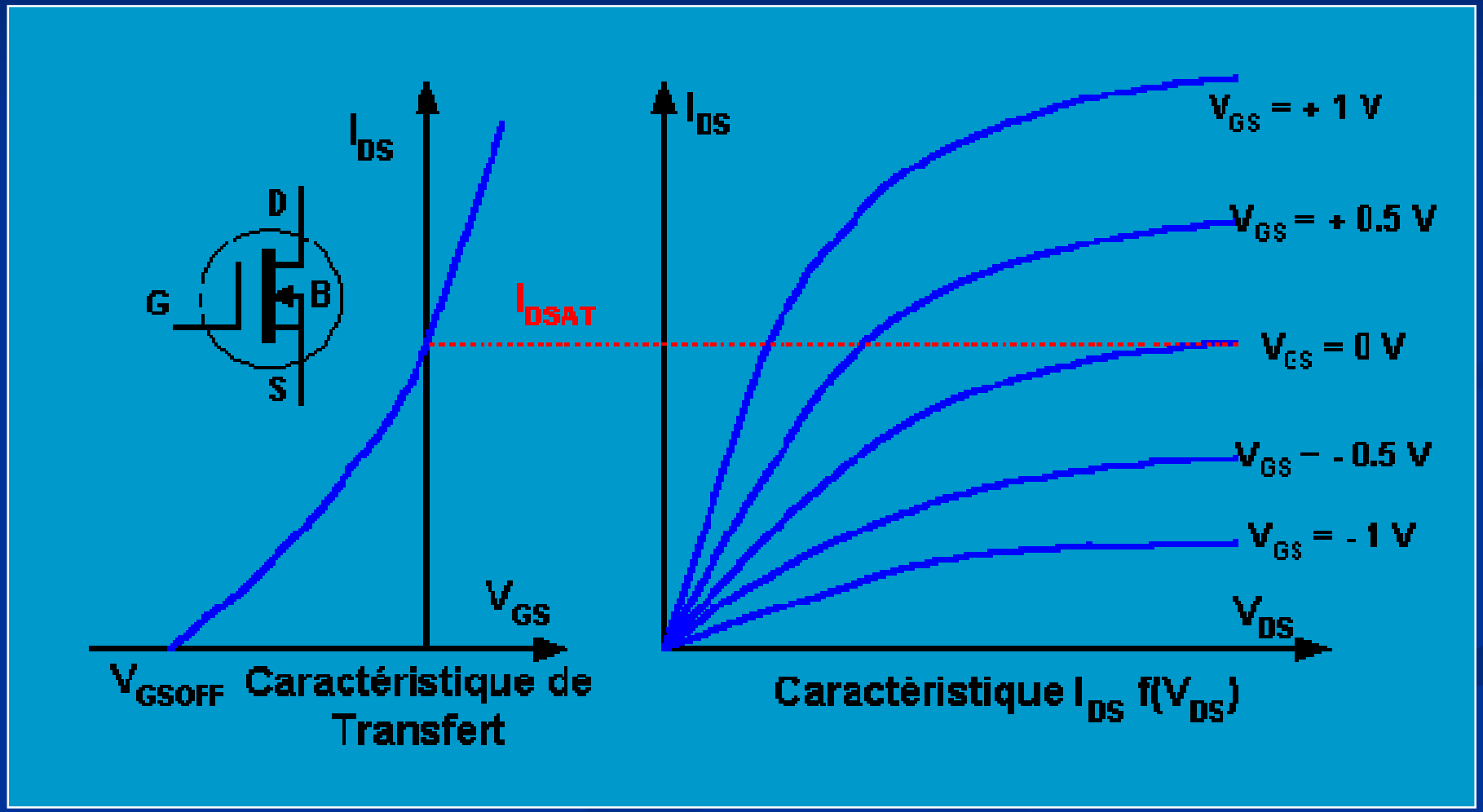
- Le transistor MOS se comporte comme un interrupteur « presque » parfait
- MOS : Metal Oxyde Semiconductor



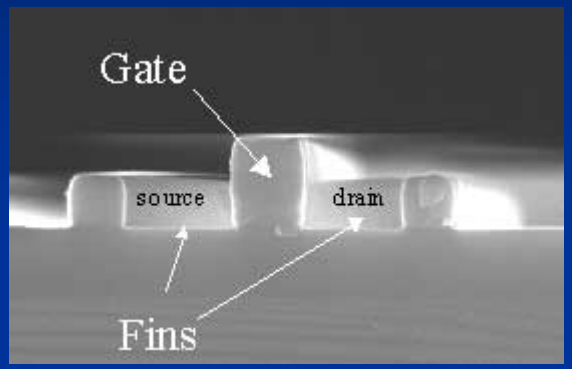


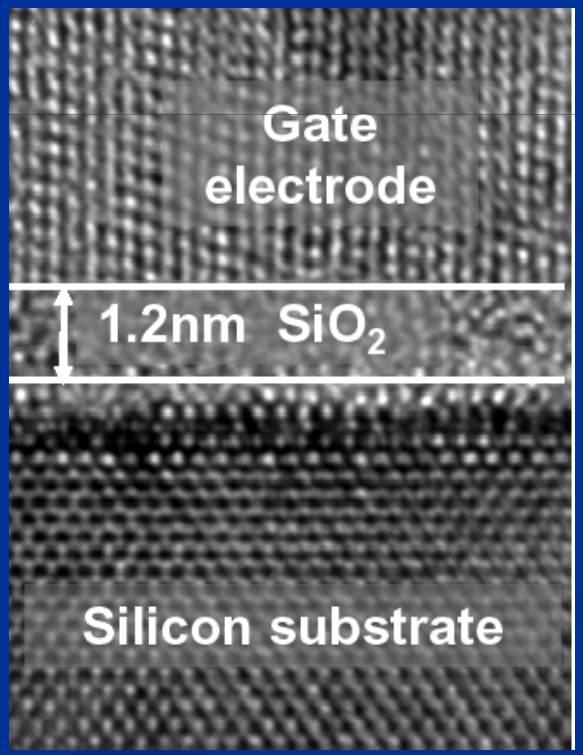


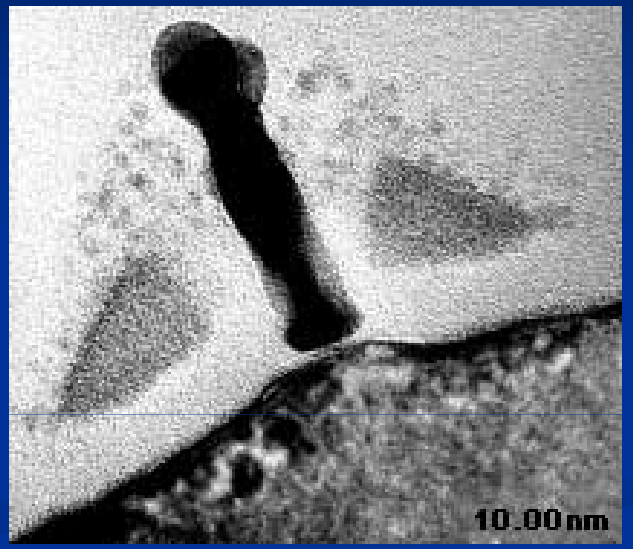
Caractéristiques MOS



Coupe MOS


Gate
source
drain
Fins

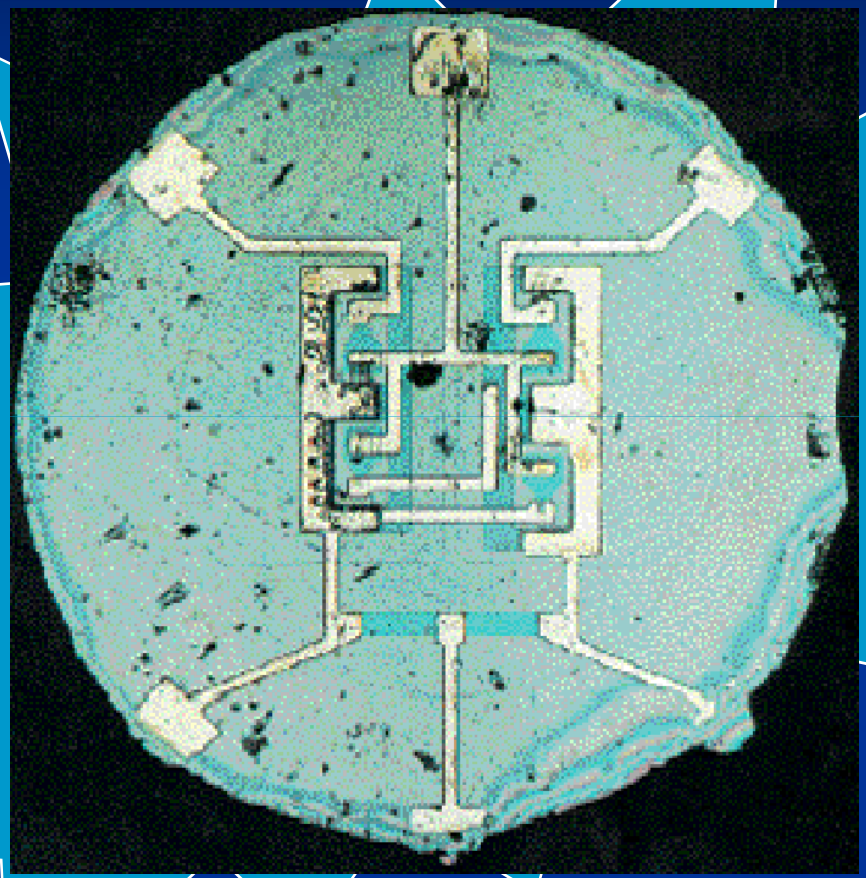

Gate
electrode
1.2nm SiO2
Silicon substrate


10.00nm

Premier Circuit Integre 1958

- Texas Instruments fabrique le premier CI :
- un inverseur à deux transistors

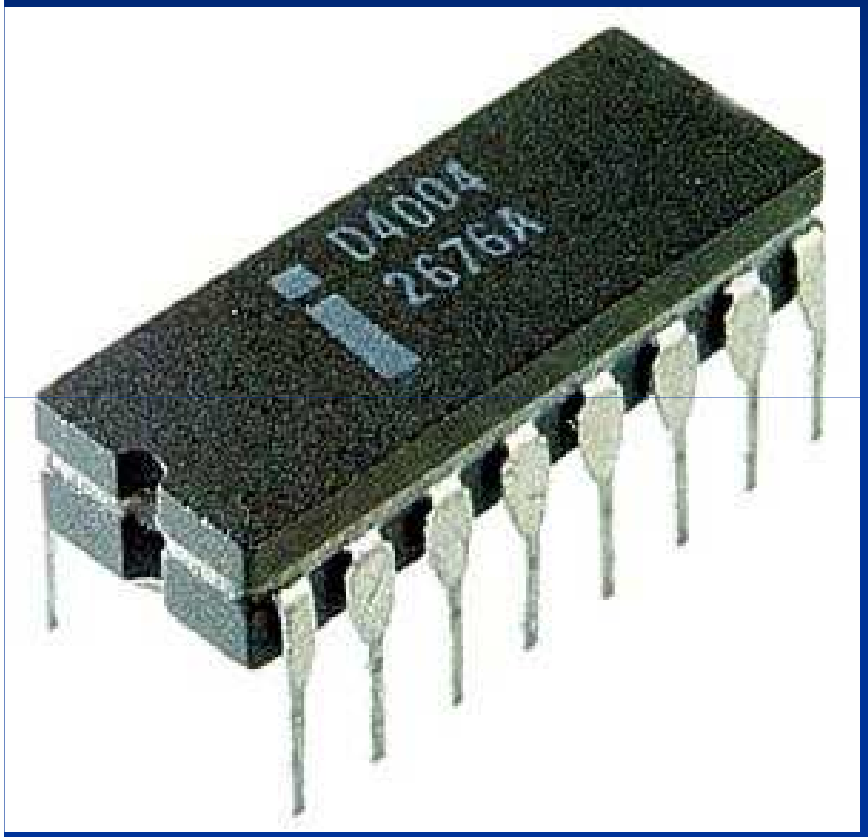
Explosion technologique : 1961

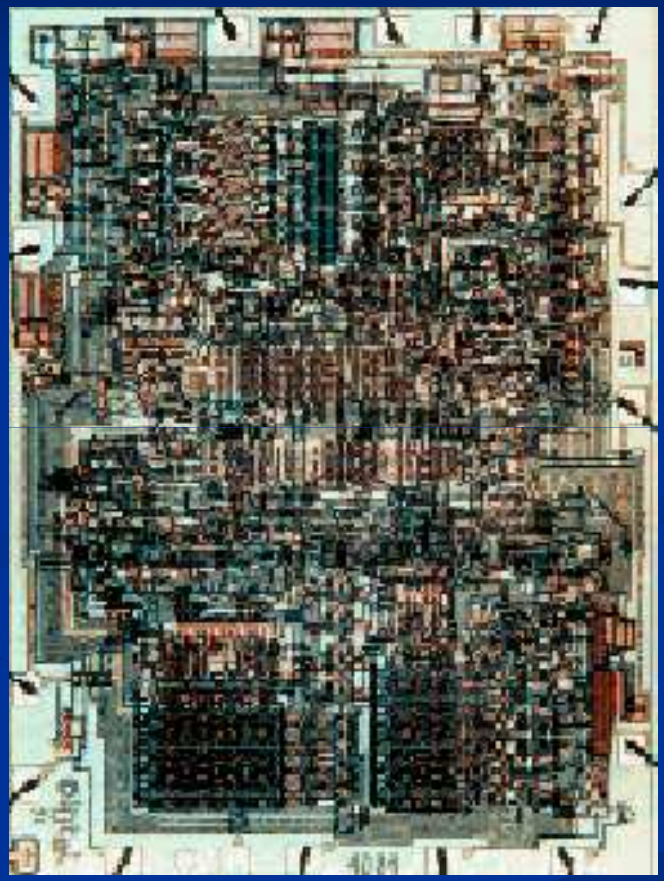


inverseur

4 transistors

INTEL 4004 le premier uProcesseur : 1971





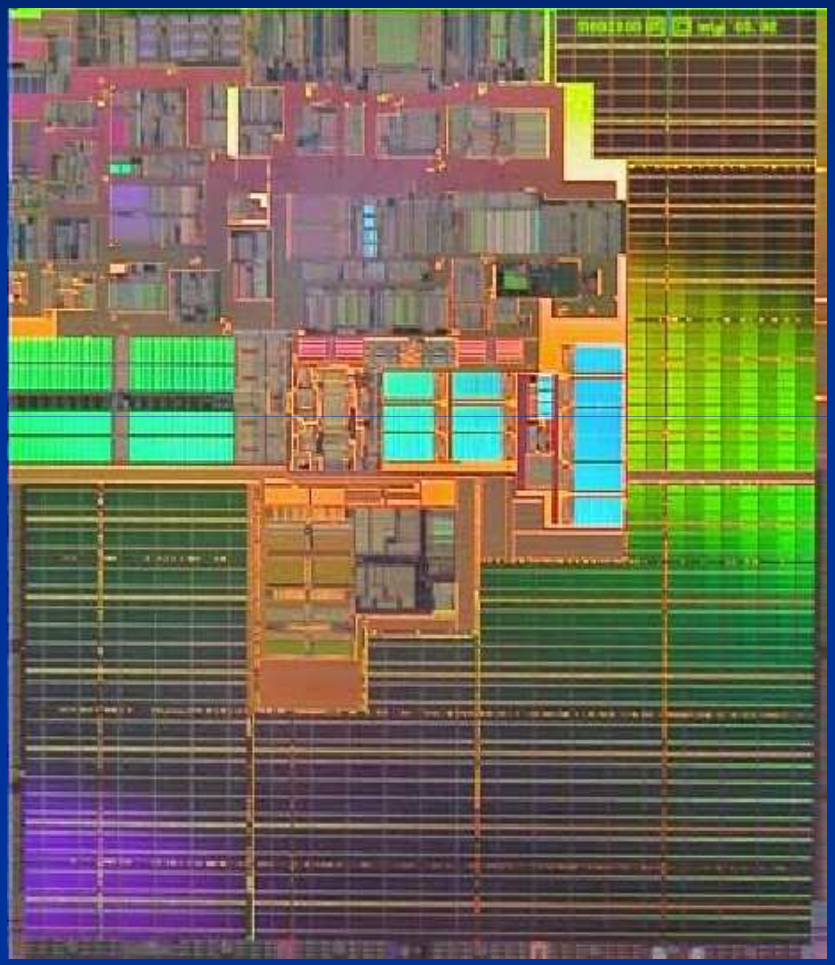
15 novembre 1971
108KHZ **2300** Transistors (valeur actuelle $20000)
technologie 10um (10000nm)

2002

2007 PENTIUM 4

44 Millions
de transistors

2008



Processeur INTEL ITANIUM

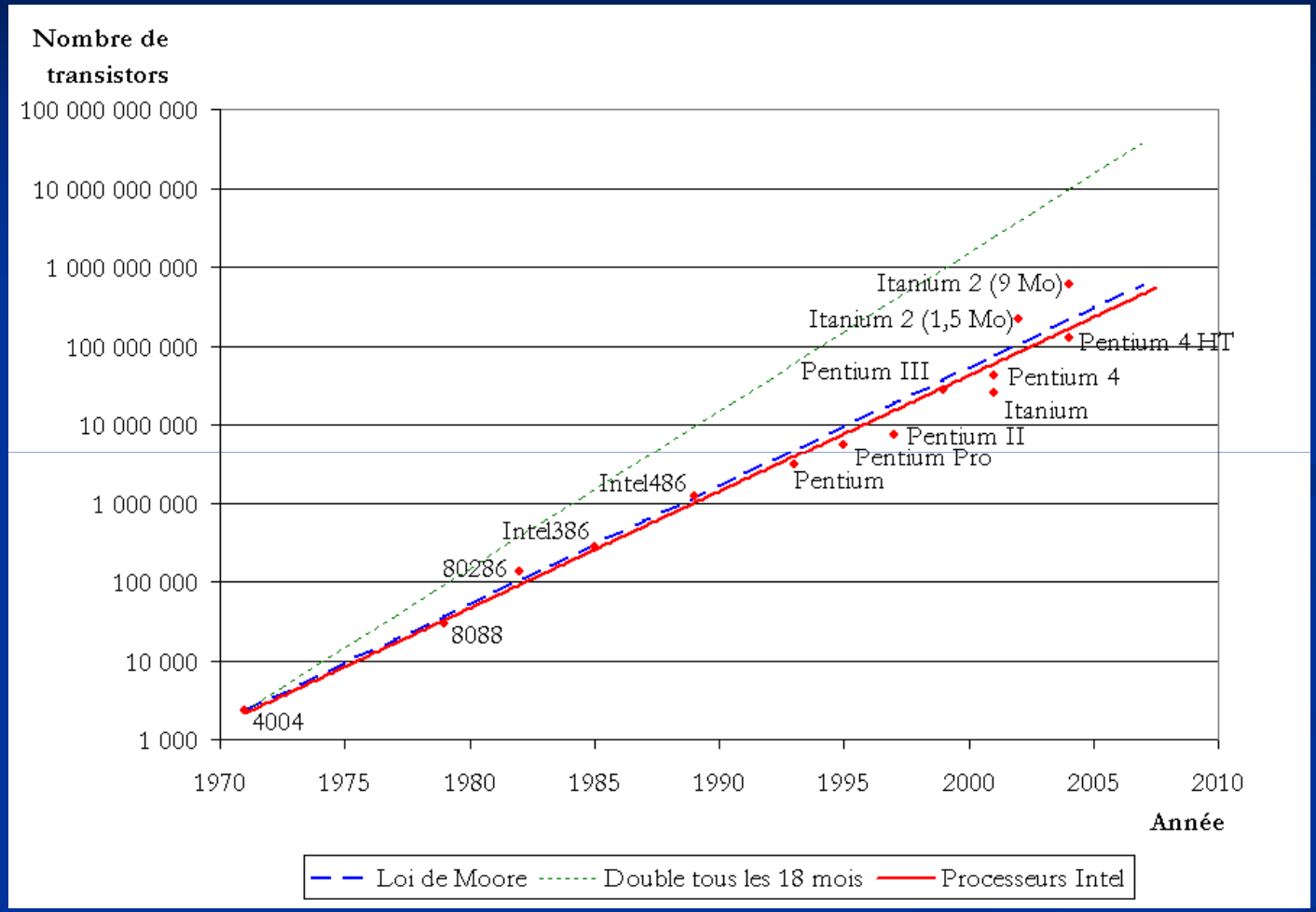
Process 130nm

6MB de RAM CACHE

410 Millions de transistor

Cheveux humain 100 µm
Virus Amoeba 15 µm
Globule rouge 7 µm
Virus du SIDA 100 nm

La loi de Moore



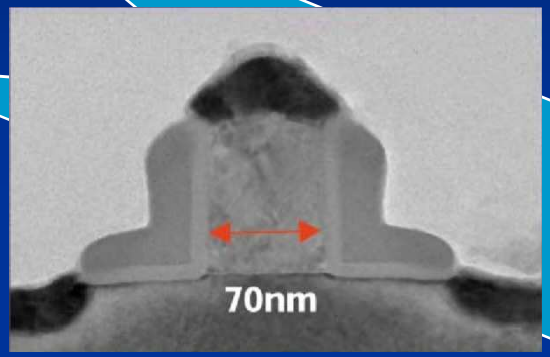
Le nombre de transistors intégrés sur une même surface double tous les 24 mois

Technologie

Caractéristiques	1999	2001	2004	2008
Process (nm) Drain-Source	180	130	90	60
Transistors par puce (Millions)	24	48	135	539
Fréquences de bus	1200	1600	2000	2700
Nombres de couches	9	7	8	9

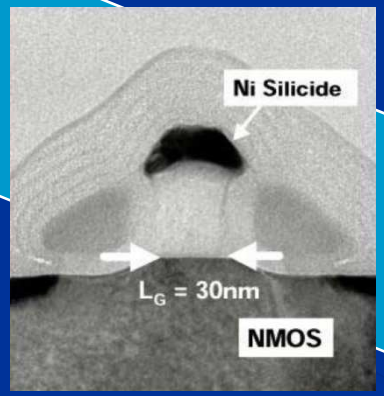
Evolution

70nm



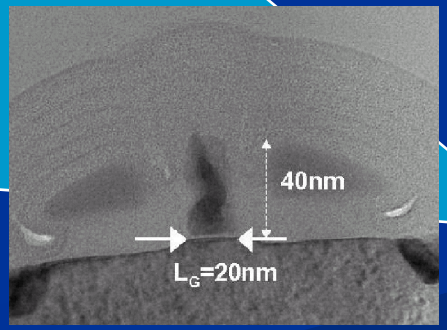
2004

30nm



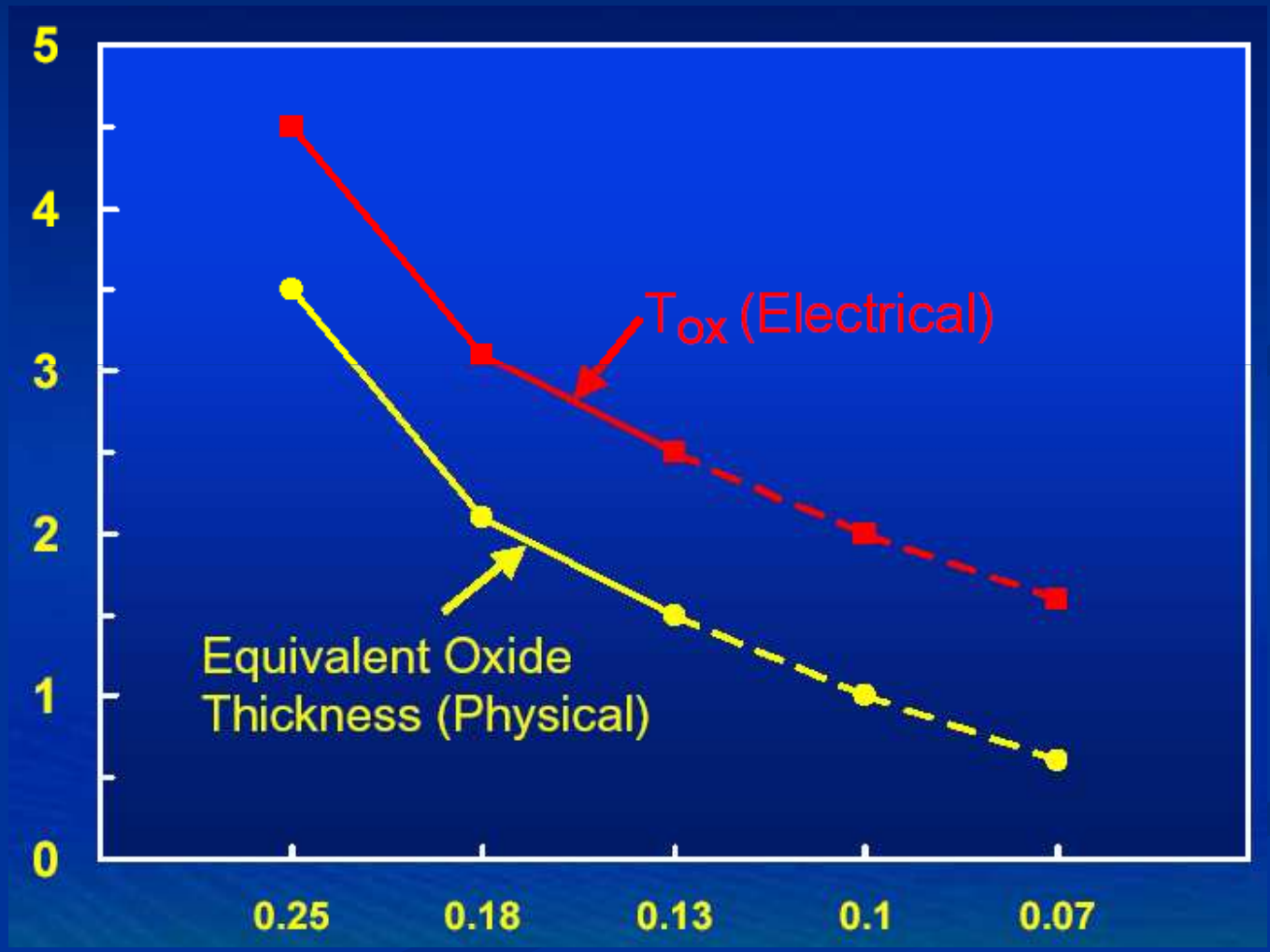
2007

20nm



2009

Epaisseur de grille / technologie



Epaisseur de grille

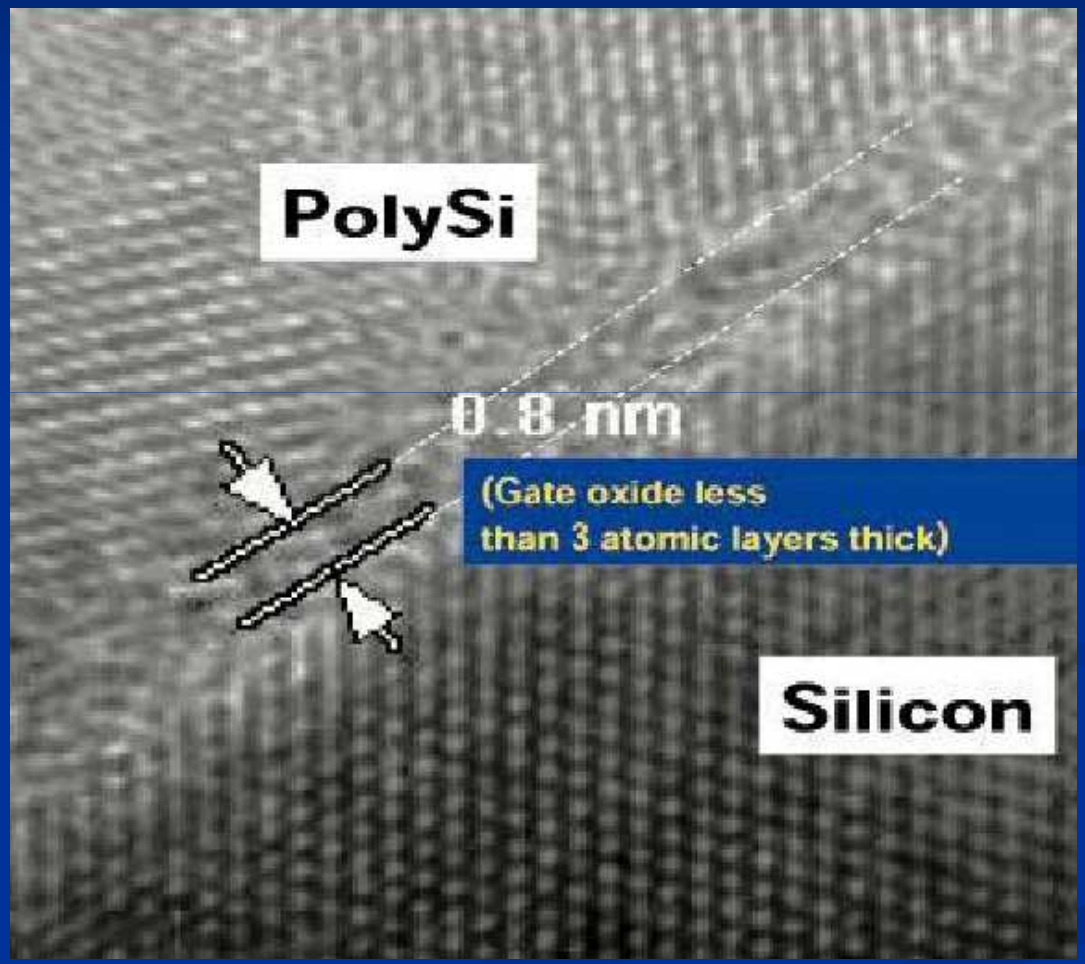
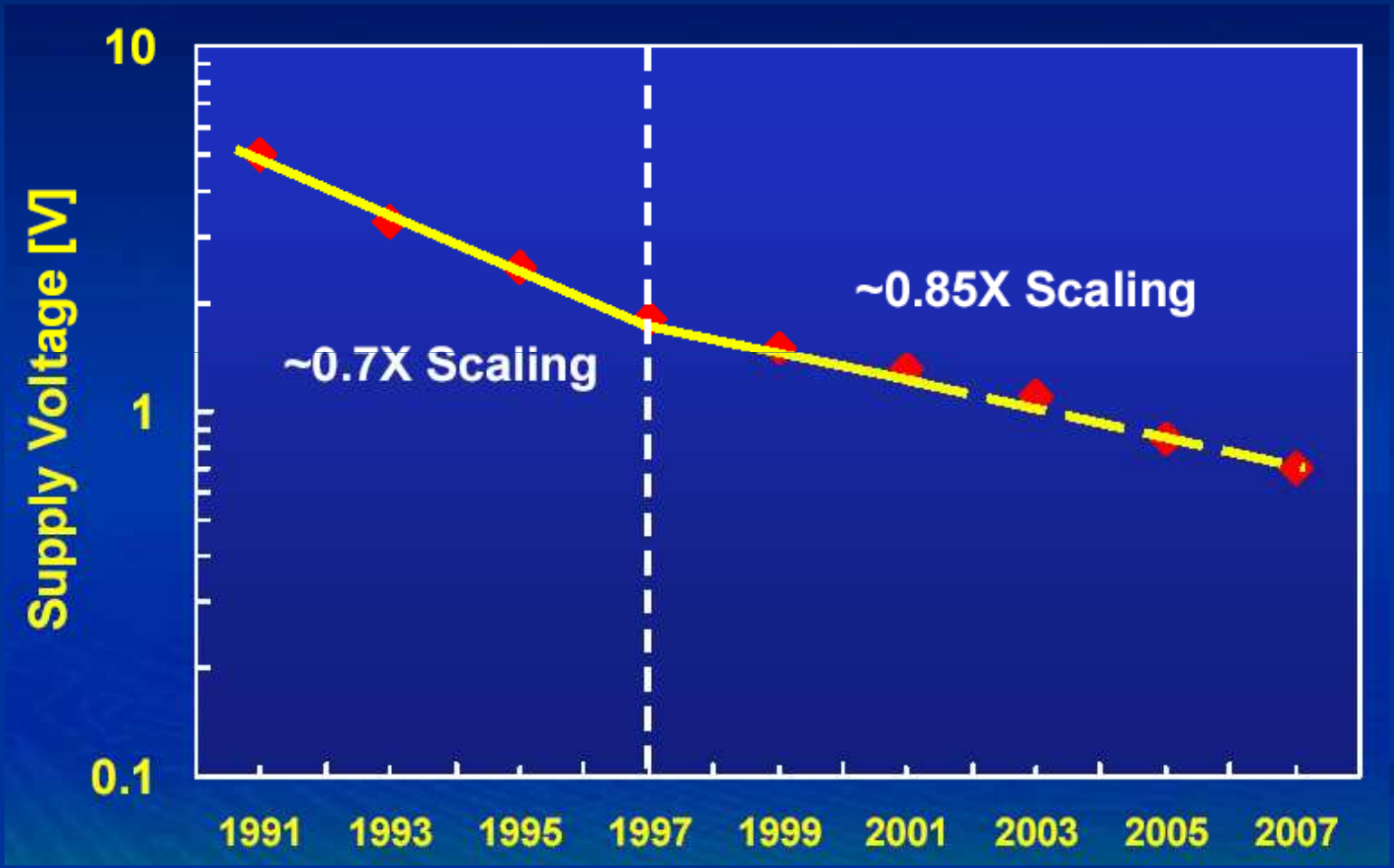


Image au microscope à effet tunel

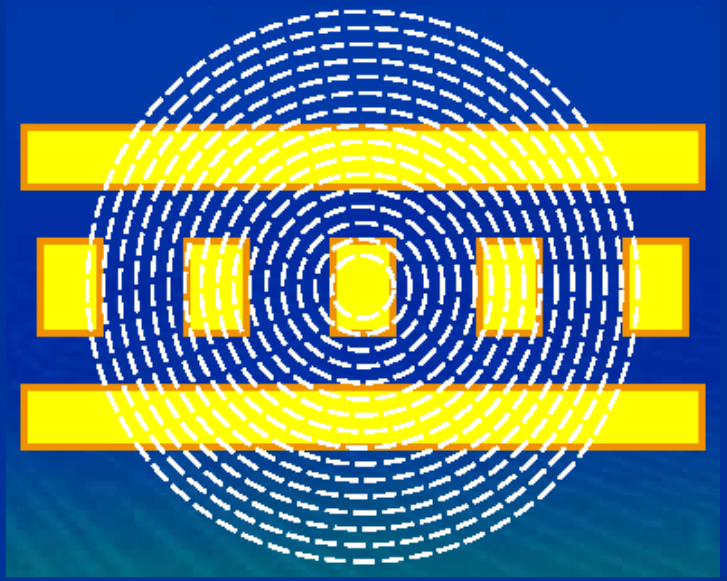
Tension d’alimentation



CEM

Couplage capacitif :

Du au champ électrique entre deux conducteur de potentiels différents



Couplage inductif :

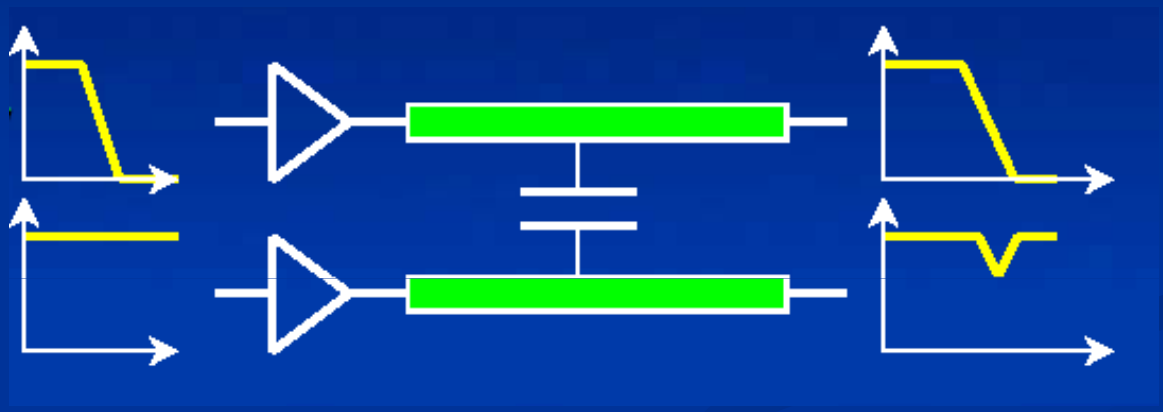
Du aux variations du champ magnétique lié aux variations de courant

Dépend de la fréquence

Les effets de la CEM

Agresseur

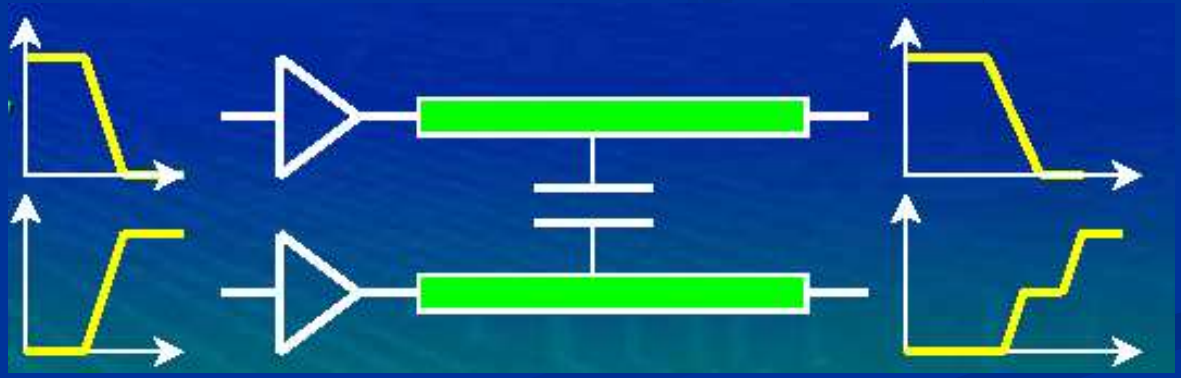
Victime



BRUIT

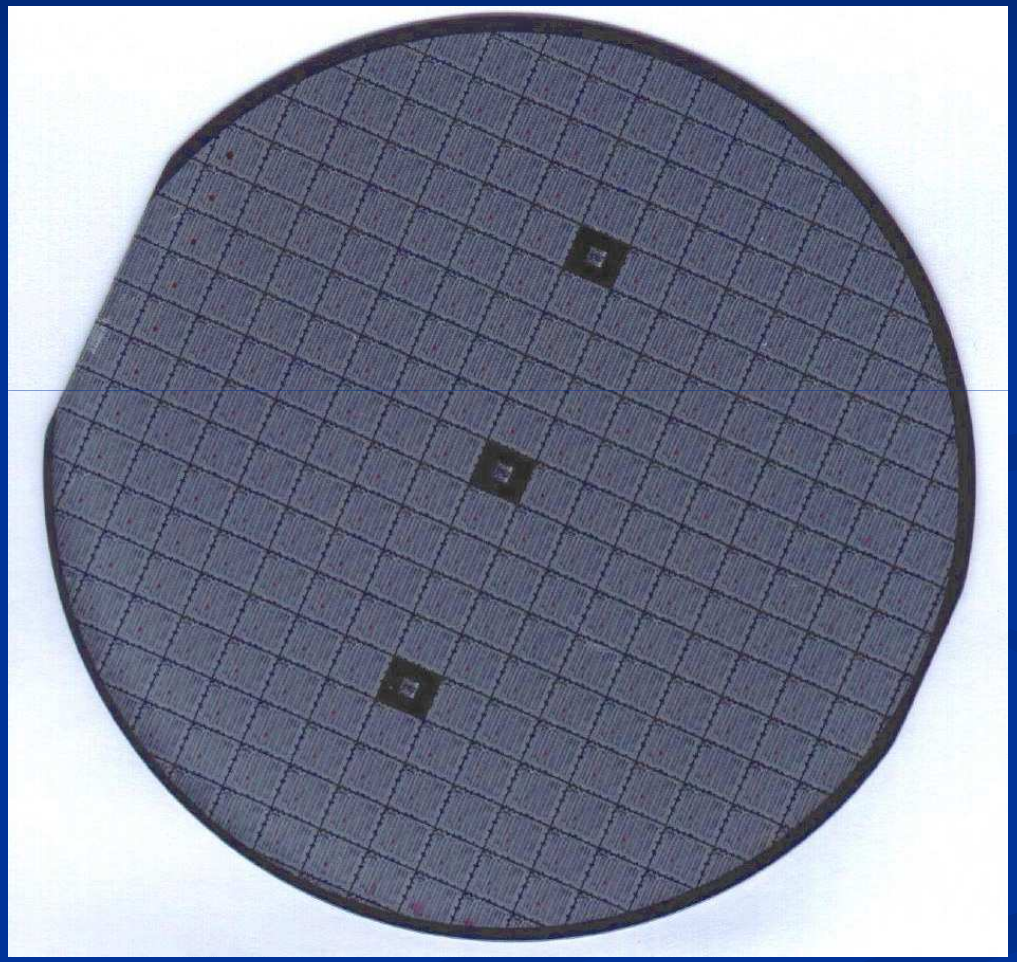
Agresseur

Victime



RETARD

Fabrication de masse

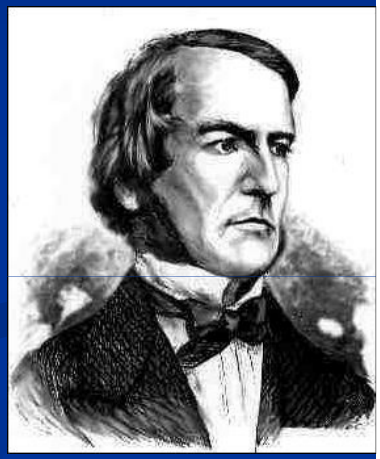


Coût des équipements

Lithoraphie par Ultra Violets

Technologie et calcul binaire

- L'évolution technologique a permis la mise en œuvre de l'algèbre de Georges Boole (1815-1864) à travers la logique combinatoire et séquentielle.



- Construction de machines d'états (2ième guerre mondiale) puis du premier micro-processeur (INTEL)

Utilisation électronique du calcul binaire

- Codage électronique des nombres avec deux états
- Réalisation d'opérations complexes à partir d'opérateurs Booléens simples (NON, ET, OU) réalisés avec des transistors.

Ecrire des nombres avec des BIT l'octet :

Bit	7	6	5	4	3	2	1	0
Val	2_7	2_6	2_5	2_4	2_3	2_2	2_1	2_0
Déc	128	64	32	16	8	4	2	1

$10110110 = 1*2_7+0*2_6+1*2_5+1*2_4+0*2_3+1*2_2+1*2_1+0*2_0$

$= 128+32+16+4+1= 181$

10110110 = B6

1011	0110
B	6

Notation hexadécimale :

$B6 ou B6h ou 0xB6

Codage

- Les nombres peuvent être écrits en :
  - Décimal : 123
  - Hexadécimal : 7Bh ou 0x7B
  - Binaire : 01111011b
  - BCD (binary coded decimal).
    - 23 en BCD : 0010 0011 (2 et 3)

Dec	Hexa
0	0
1	1
2	2
3	3
4	4
5	5
6	6
7	7
8	8
9	9
10	A
11	B
12	C
13	D
14	E
15	F

8 bits représentent un octet (byte)
16 bits représentent un mot (word)
32 bits représentent un mot long (long)

Pour représenter les nombres signés (positifs ou négatifs) on utilise le bit de poids fort
ex :non signé un octet est compris entre 0 et 255
signé il est compris en -128 et +127

Binaire signé	Dec
00000011	3
00000010	2
00000001	1
00000000	0
11111111	-1
11111110	-2
11111101	-3
11111100	-4
111110101	-5

ASCII

- *L'American Standard Code for Information Interchange* permet l'échange de caractères alphanumériques (lettres, chiffres, ponctuation, contrôles)

code		1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0x00	NUL	SOH	STX	ETX	EOT	ENQ	ACK	BEL	BS	HT	LF	VT	NP	CR	SO	SI
0x10	DLE	DC1	DC2	DC3	DC4	NAK	SYN	ETB	CAN	EM	SUB	ESC	FS	GS	RS	US
0x20	SP	!	"	#	$	%	&	'	(	)	*	+	,	-	.	/
0x30	0	1	2	3	4	5	6	7	8	9	:	;	<	=	>	?
0x40	@	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
0x50	P	Q	R	S	T	U	V	W	X	Y	Z	[	\	]	^	_
0x60	`	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o
0x70	p	q	r	s	t	u	v	w	x	y	z	{	|	}	~	DEL

Attention, le chiffre '5' codé en ASCII vaut 35h, ce qui n'a rien à voir avec la valeur 5

Faire des OPERATIONS
EX : ADDITION BINAIRE

$$\begin{array}{r} {\scriptstyle 1\ \ 1\ \ \ \ } \\ 1011 \\ +\ 1110 \\ \hline 11\,001 \end{array}$$

-X : rendre un nombre négatif
Le complément à2

5 = 00000101

Inversion ! 11111010

+1 -5 = 11111011

Inversion ! 00000100 +1 00000101

Un exemple : -3+5

11111101 (-3)

\+ 00000101 (5)

= 100000010 (2)

RETENUE

Fonction logiques

Les opérations sont réalisables à partir de fonctions logiques

OU OUI NON
OU-EXCLUSIF

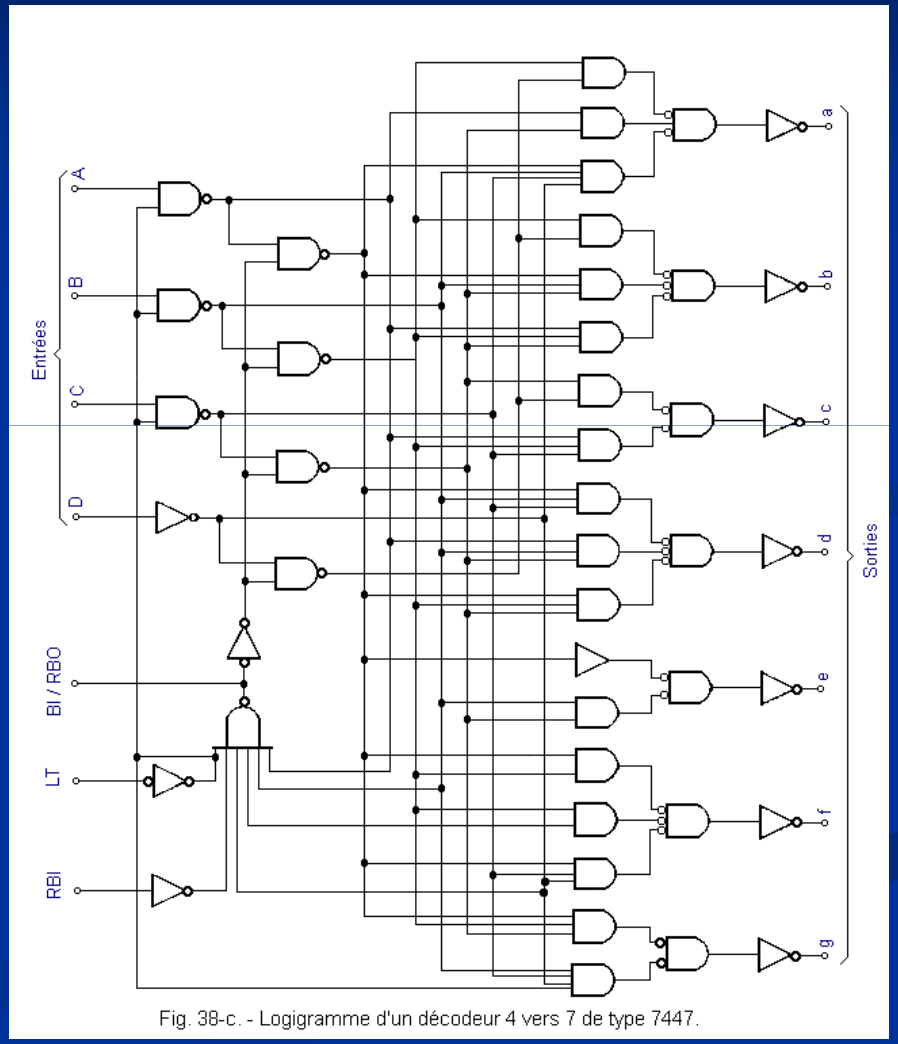


Fig. 38-c. - Logigramme d'un décodeur 4 vers 7 de type 7447.

NON

entree	Sortie
0	1
1	0

PORTA	x	x	x	x	x	x	x	x
! PORTA	/x	/x	/x	/x	/x	/x	/x	/x

AND : &

Bit1	Bit2	AND
0	0	0
0	1	0
1	0	0
1	1	1

Masquage

PORTA	x	x	x	x	x	x	x	x
&	0	0	0	1	0	0	0	0
=	0	0	0	x	0	0	0	0

Forçage à 0

PORTA	x	x	x	x	x	x	x	x
&	1	1	1	0	1	1	1	1
=	x	x	x	0	x	x	x	x

OR : |

Bit1	Bit2	OR
0	0	0
0	1	1
1	0	1
1	1	1

Forçage à 1

PORTA	x	x	x	x	x	x	x	x
OU	0	0	0	1	0	0	0	0
=	x	x	x	1	x	x	x	x

XOR : ^

Bit1	Bit2	XOR
0	0	0
0	1	1
1	0	1
1	1	0

BASCULEMENT

PORTA	x	x	x	x	x	x	x	x
XOR	0	0	0	1	0	0	0	0
=	x	x	x	/x	x	x	x	x

DECALAGE : <<

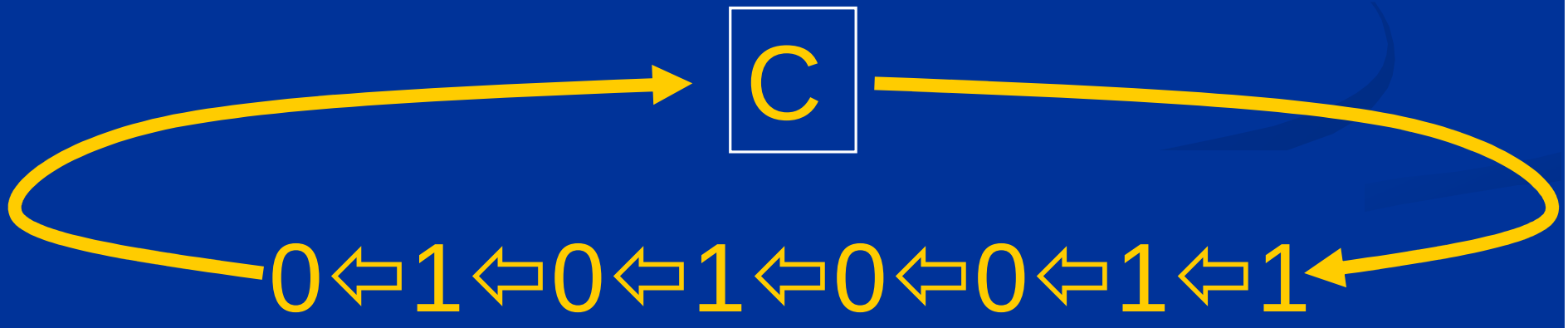
0 1 0 1 0 0 1 1

 ⇦0⇦1⇦0⇦1⇦0⇦0⇦1⇦1⇦0

1 0 1 0 0 1 1 0

ROTATION

0 1 0 1 0 0 1 1



1 0 1 0 0 1 1 C

QUIZZ

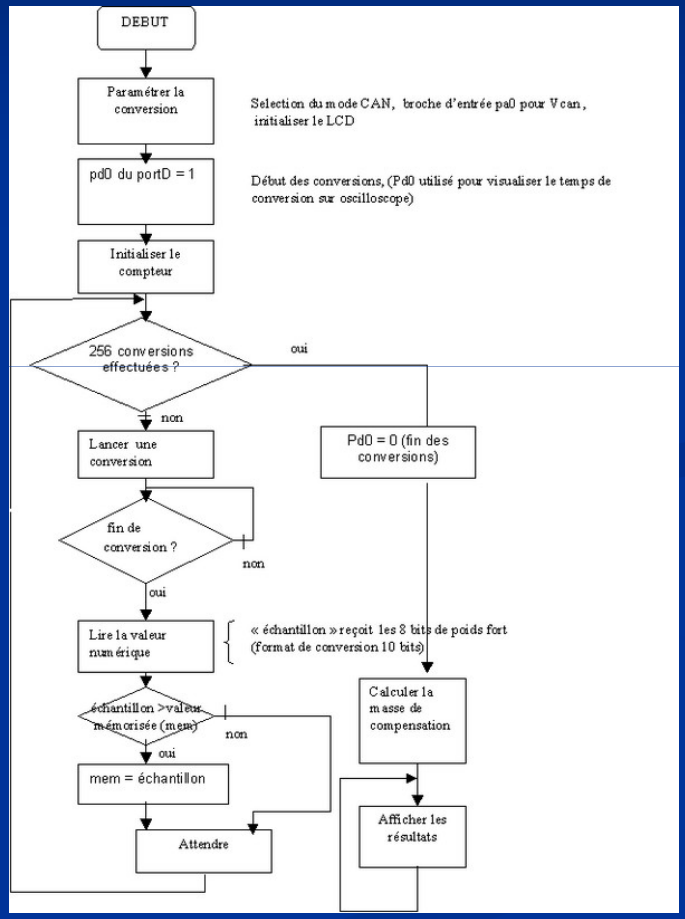
QUIZZ

- 5Ah + 78h = ( ) hexa
- 10011010 − 01111000 — Interpréter le résultat suivant que les nombres soient signés ou non ?
- 10110001 + 10001101 — Y a t-il overflow ? comment y palier ?
- Addition en code BCD résultat en décimal et BCD : 14 + 89
- Combien de bits pour représenter le nombre 1492 ?
- Un double mot est constitué de combien de bits ?
- Combien de koctets représente une mémoire de 256 kbits ?
- Combien de cases mémoires possède une mémoire de 64 koctets ?
- Intérêt du code hexadécimal ?
- Dans une machine à laver, quel composant est le mieux adapté: un microprocesseur, un microcontrôleur ou un DSP ?
- Dans un analyseur de spectre, quel composant est le mieux adapté ?
- Pour commander un moteur, quel composant est le mieux adapté ?
- Quels sont les avantages des microcontrôleurs
- Donnez un exemple d'utilisation de la fonction XOR
- Quel est l'ensemble des nombres entiers signés codés sur 16bits

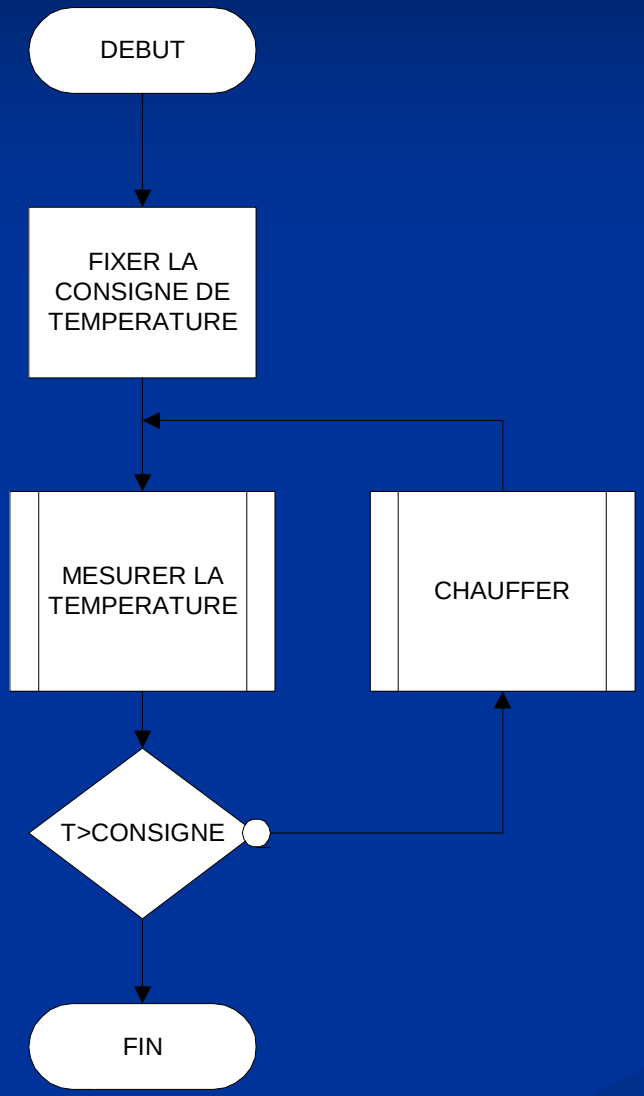
QUIZZ

- Qu ’appelle-t-on un système embarqué ?
- Qu ’appelle-t-on un microcontrôleur 8 bits ?
- Qu ’appelle-t-on un périphérique ?
- Pourquoi les fréquences d ’horloge des microcontrôleurs sont-elles relativement faible ?
- Quelle est l’ordre de grandeur de l’horloge de cadencement d’un microcontrôleur ?
- Comment choisit-on un microcontrôleur ?
- Quelles différences faites vous entre un microprocesseur et un microcontrôleur ?

Eléments d'algorithmie

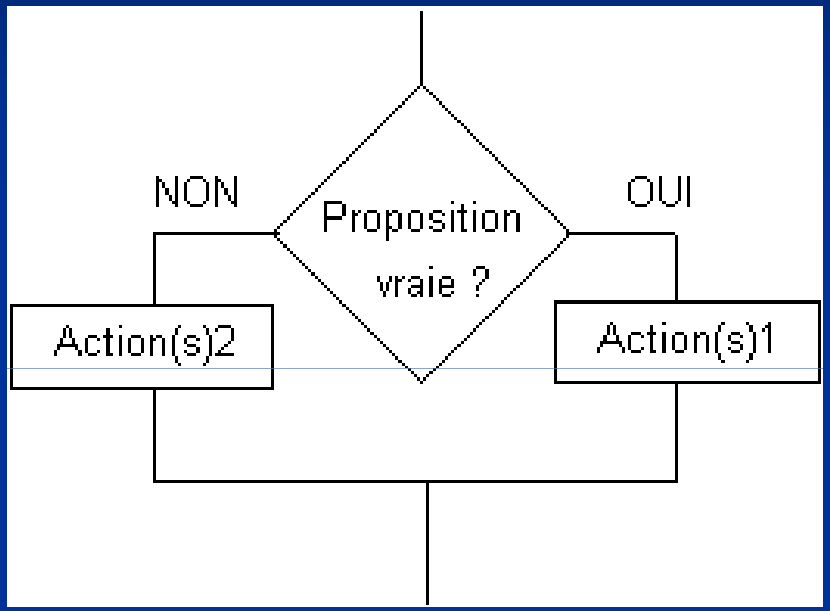


Algorigramme / Algorithme

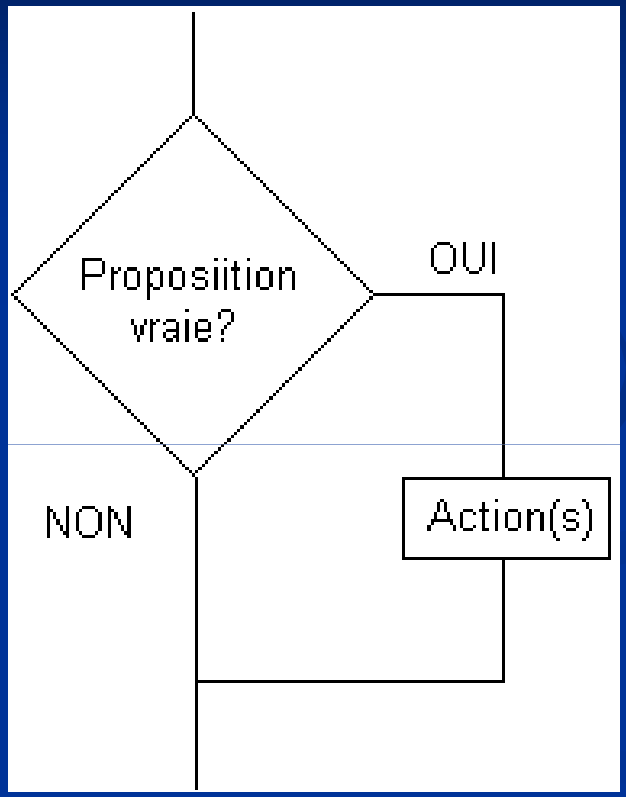


Debut
Fixer la température de consigne
Tant que T<Consigne
Chauffer
Fin Tant que
Fin

Structures alternatives

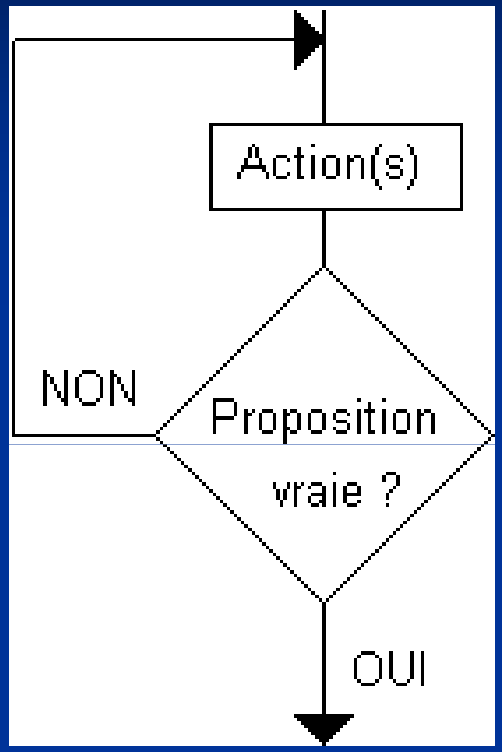


SI *propositionvraie*
alors *action(s)1*
sinon *action(s)2*
Fin si

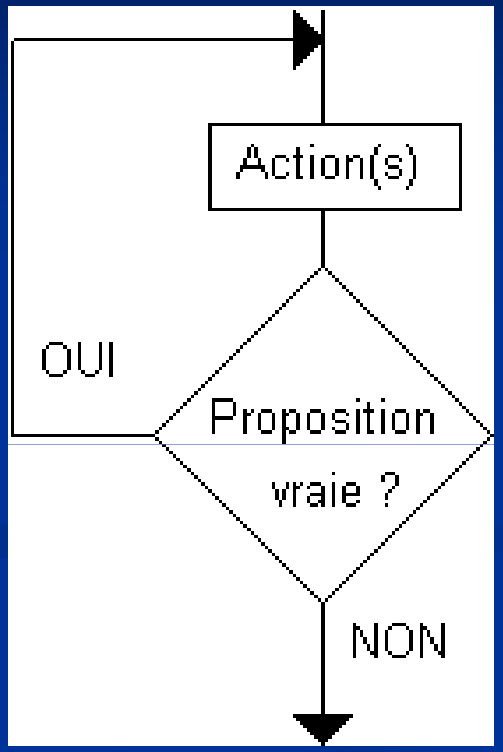


SI *proposition vraie*
alors *action(s)*
Fin si

Structures répétitives



Répéter *action(s)*
jusqu'à *proposition vraie*



Répéter *action(s)*
tant que *proposition vraie*

Langages

C

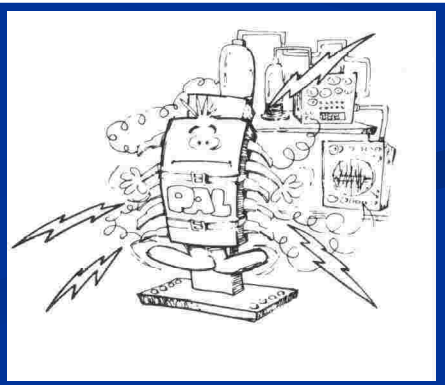
```
char heure,min,sec;
int pression,temp;
char* calcul(int m)
{ char c;
   m=valeur/1000;
   c=(valeur-(1000*m))/100;
   if (c>10) return (c );
}
```

ASSEMBLEUR

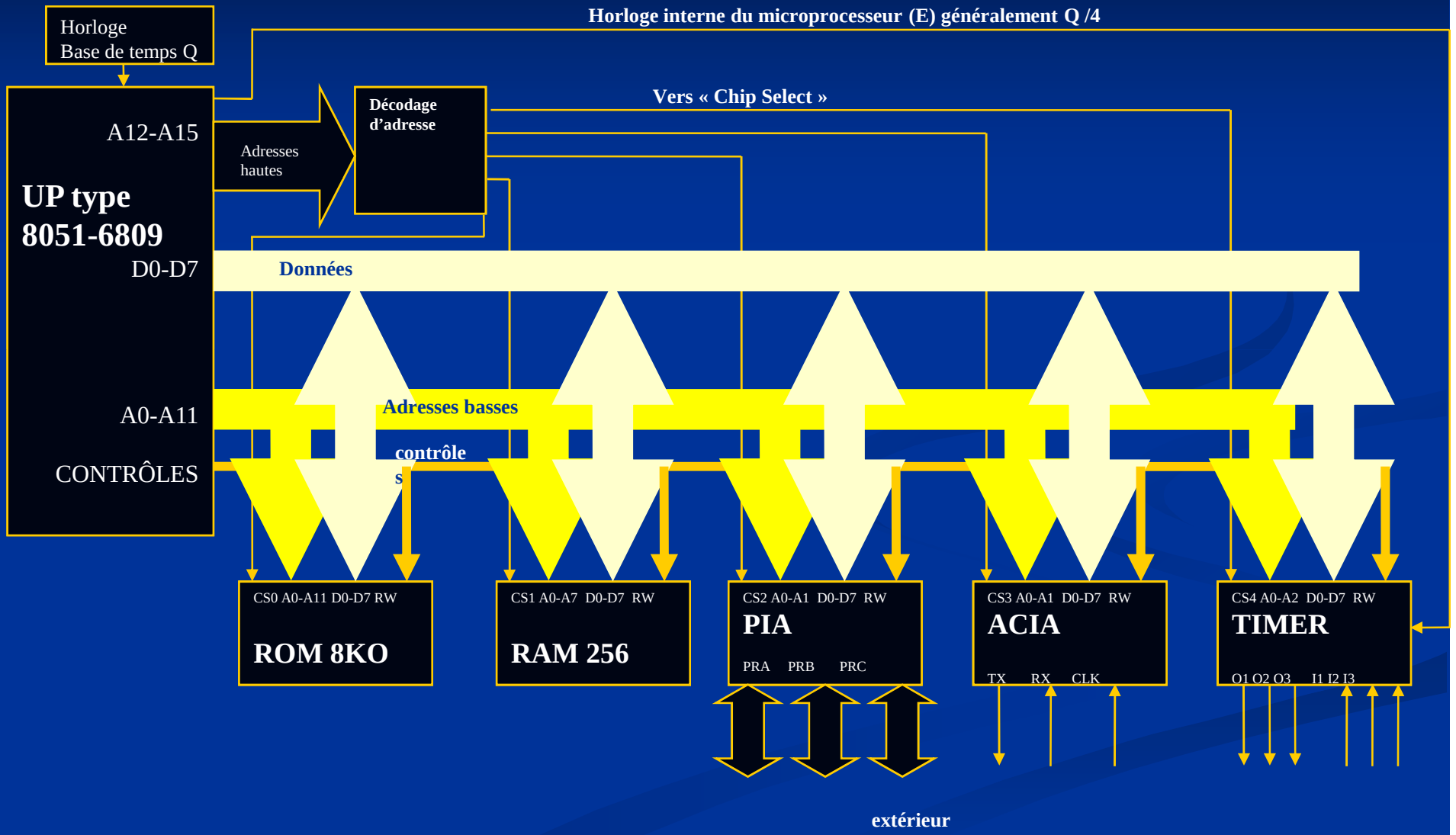
```
LED EQU    P5.0
CSEG   AT 0
MOV SP,#7Fh   ;
SUITE: CPL LED
CALL   TEMPO
SJMP   SUITE
TEMPO: MOV R0,#0FFh
TEMPO1:MOV    R1,#0FFh
TEMPO2:DJNZ   R1,TEMPO2
 DJNZ  R0,TEMPO1
 RET
```

Langage machine:

```
0x0000   75817F
0x0003   B2A0
0x0005   1109
0x0007   80FA
0x0009   78FF
0x000B   79FF
0x000D   D9FE
0x000F   D8FA
0x0011   22
```



Système minimum



Décodage d'adresses

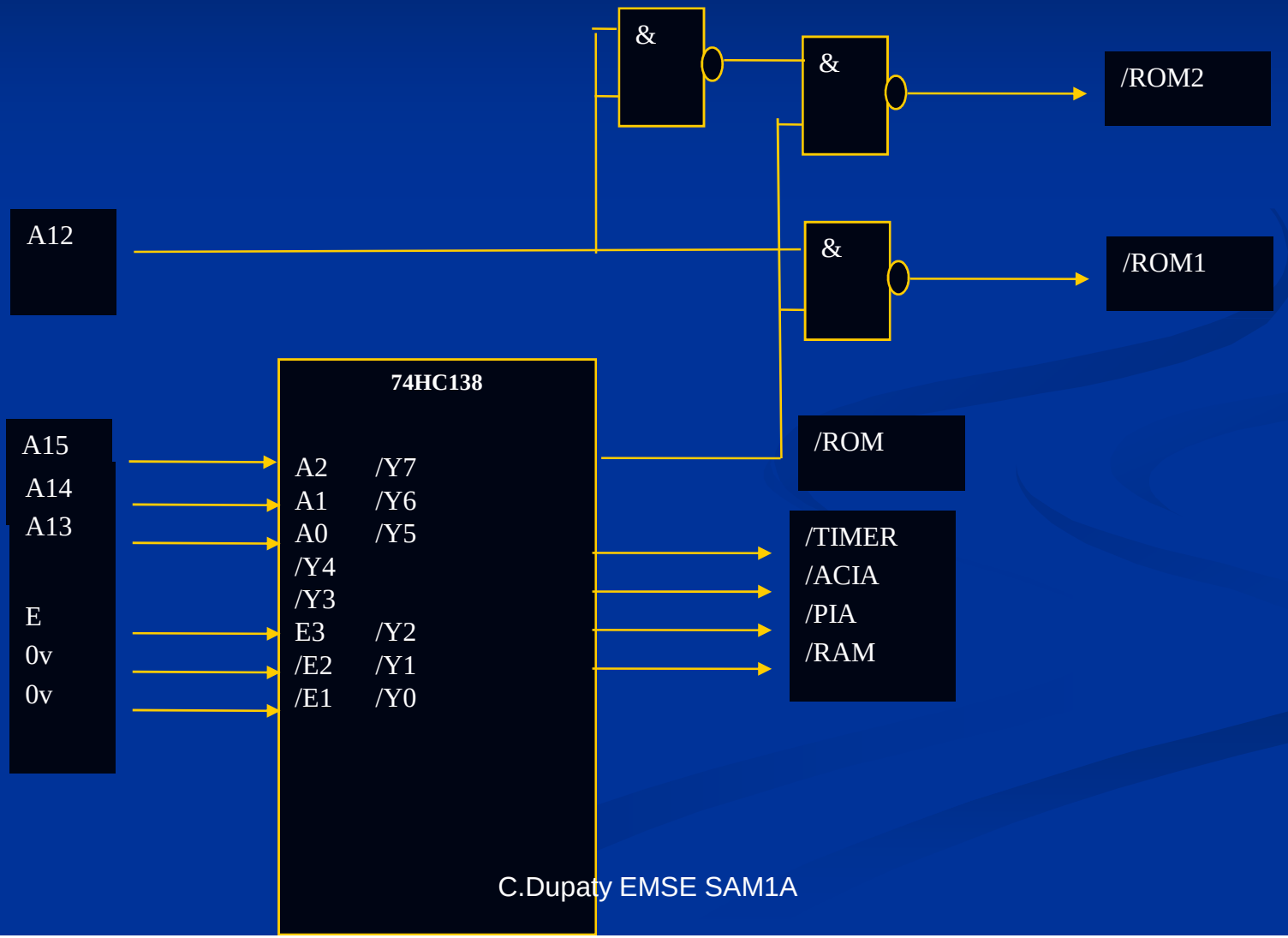
$FFFF $F000	PROM1 4KO
$EFFF $E000	PROM2 4KO

32K	16K	8K	4K	2K	1K	512	256	128	64	32	16	8	4	2	1		
A15	A14	A13	A12	A11	A10	A9	A8	A7	A6	A5	A4	A3	A2	A1	A0	Hexadécimal	/CS
1	1	1	1	x	x	x	x	x	x	x	x	x	x	x	x	$F000-$FFFF	ROM1
1	1	1	0	x	x	x	x	x	x	x	x	x	x	x	x	$E000-$FFFF	ROM2
1	0	1	*	*	*	*	*	*	*	*	*	*	x	x	x	$A000-$A007	TIMER
0	1	1	*	*	*	*	*	*	*	*	*	*	*	x	x	$6000-$6003	ACIA
0	1	0	*	*	*	*	*	*	*	*	*	*	*	x	x	$4000-$4003	PIA
0	0	0	*	*	x	x	x	x	x	x	x	x	x	x	x	$0000-$07FF	RAM

Décodage par portes logiques

- RAM=A15+A14+A13+A12
- PIA=A15+/A14+A13+A12
- ACIA=A15+/A14+/A13+A12
- TIMER=/A15+A14+/A13+A12
- ROM2=/A15+/A14+/A13+A12
- ROM1= /A15+/A14+/A13+/A12

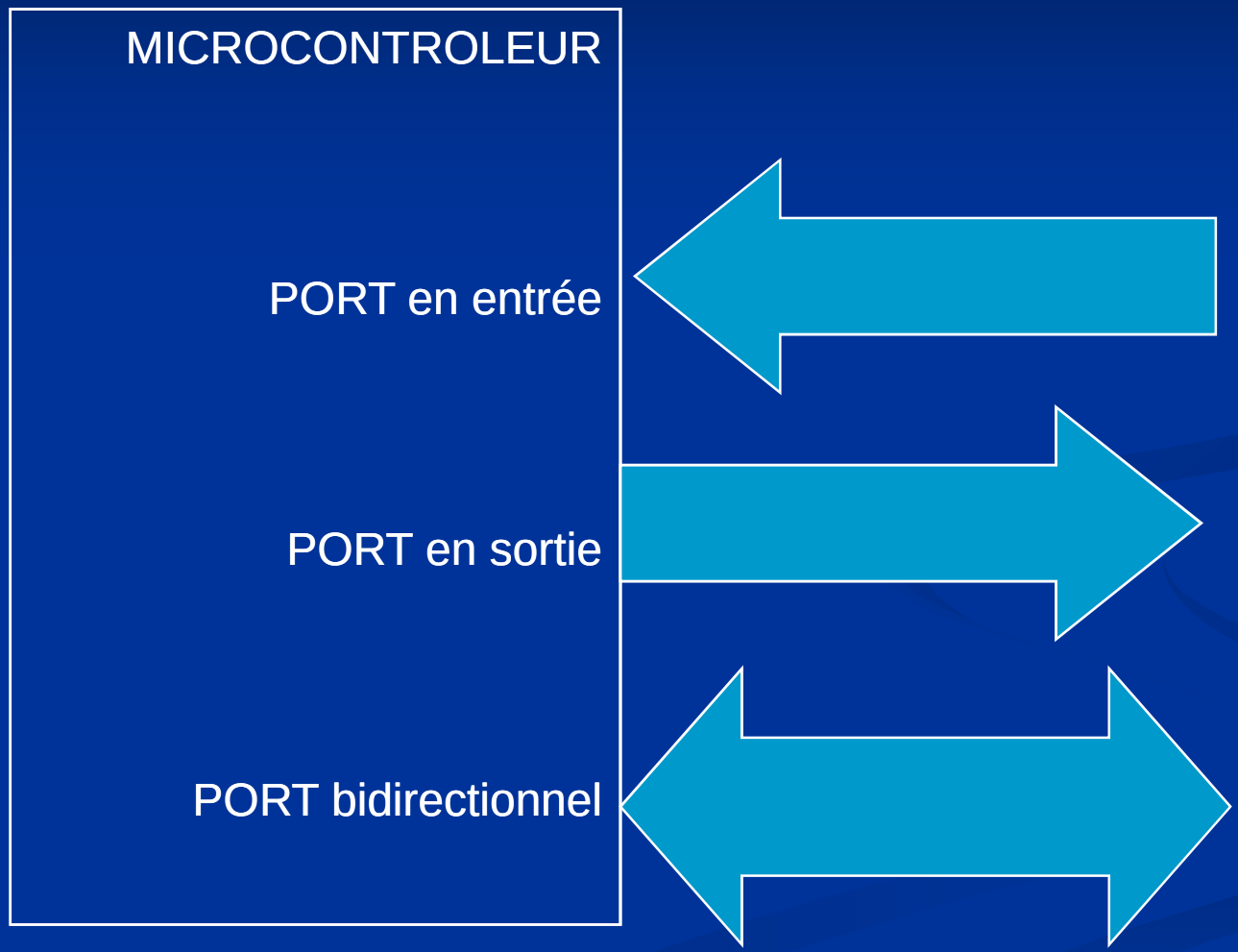
Decodage par décodeur intégré



Décodage par circuit programmable PAL

```
module DECODAD;
title 'Decodeur d adresse';
Declarations
        DECODAD  device  'P16L8';
        A15,A14,A13,A12,A11,A10 pin 1,2,3,4,5,6;
        ROM1,IO,ROM2,DRAM       pin 14,15,16,17;
        H,L,X   = 1,0,.X.;
        Address = [A15,A14,A13,A12, A11,A10,X,X, X,X,X,X,
X,X,X,X];
equations
        !DRAM   = (Address <= ^hDFFF);
        !IO     = (Address >= ^hE000) & (Address <= ^hE7FF);
        !ROM2   = (Address >= ^hF000) & (Address <= ^hF7FF);
        !ROM1   = (Address >= ^hF800);
test_vectors
            (Address -> [ROM1,ROM2,IO,DRAM])
             ^h0000  -> [  H,  H,  H,   L ];
             ^h4000  -> [  H,  H,  H,   L ];
             ^h8000  -> [  H,  H,  H,   L ];
             ^hC000  -> [  H,  H,  H,   L ];
             ^hE000  -> [  H,  H,  L,   H ];
             ^hE800  -> [  H,  H,  H,   H ];
             ^hF000  -> [  H,  L,  H,   H ];
             ^hF800  -> [  L,  H,  H,   H ];
end DECODAD
```

PORTS PARALELLES



Lexique

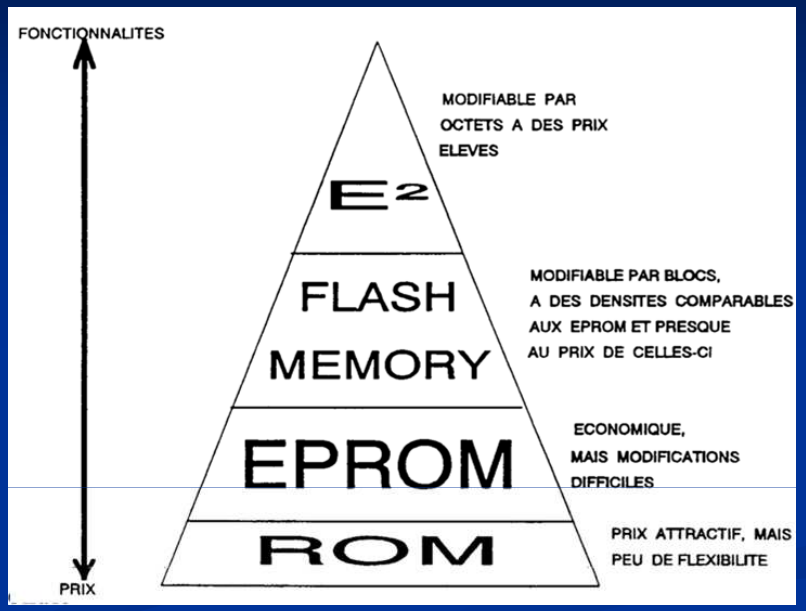
- **UP:** 8 bits, 64KO type 8051, 68HC11, PIC16, PIC18, ATMEL AVR , ST6, ST7…
- **ROM:** mémoire morte PROM ou EPROM, elle contient le programme
- **RAM** : Mémoire vive statique , elle contient les données (variables)
- **PIA, PIO :** Peripherical interface adapter, periphérical input/output. Port parallèle 8bits
- **UART , USART** : Asynchronous communications interface adapter ou Universal Synchrone/Asynchrone Receive/Transiver. Port série code NRZ .
- **TIMER** : PTM : Programmable Timer Module . Production et mesures de durées.
- **Décodeur d'adresses** : permet à partir du bus d'adresse (A0-A15) ou d'une partie de celui ci de créer des signaux permettant de sélectionner les différents boîtiers (CS pour Chip Select) lorsque l'adresse du boîtier considéré est est active.

Interfaces Internes/Externes

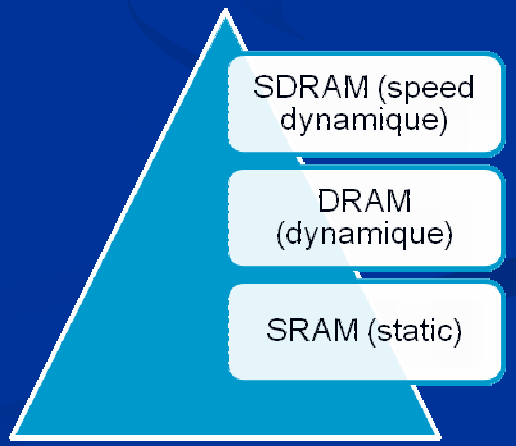
- EEPROM : mémoire morte effaçable électriquement
- ADC : convertisseurs analogiques numériques
- DAC : convertisseurs numériques analogiques
- Comparateurs de tension
- Horloge temps réel (RTC)
- Port parallèle de puissance
- Interface série synchrone SPI ou IIC
- Gestionnaire afficheur LCD graphique / alphanumérique
- Interface ethernet
- Interface WIFI
- Bus de communications de terrain (CAN ou LIN)
- Superviseur d'alimentation / Chien de garde (watch dog)

Mémoires

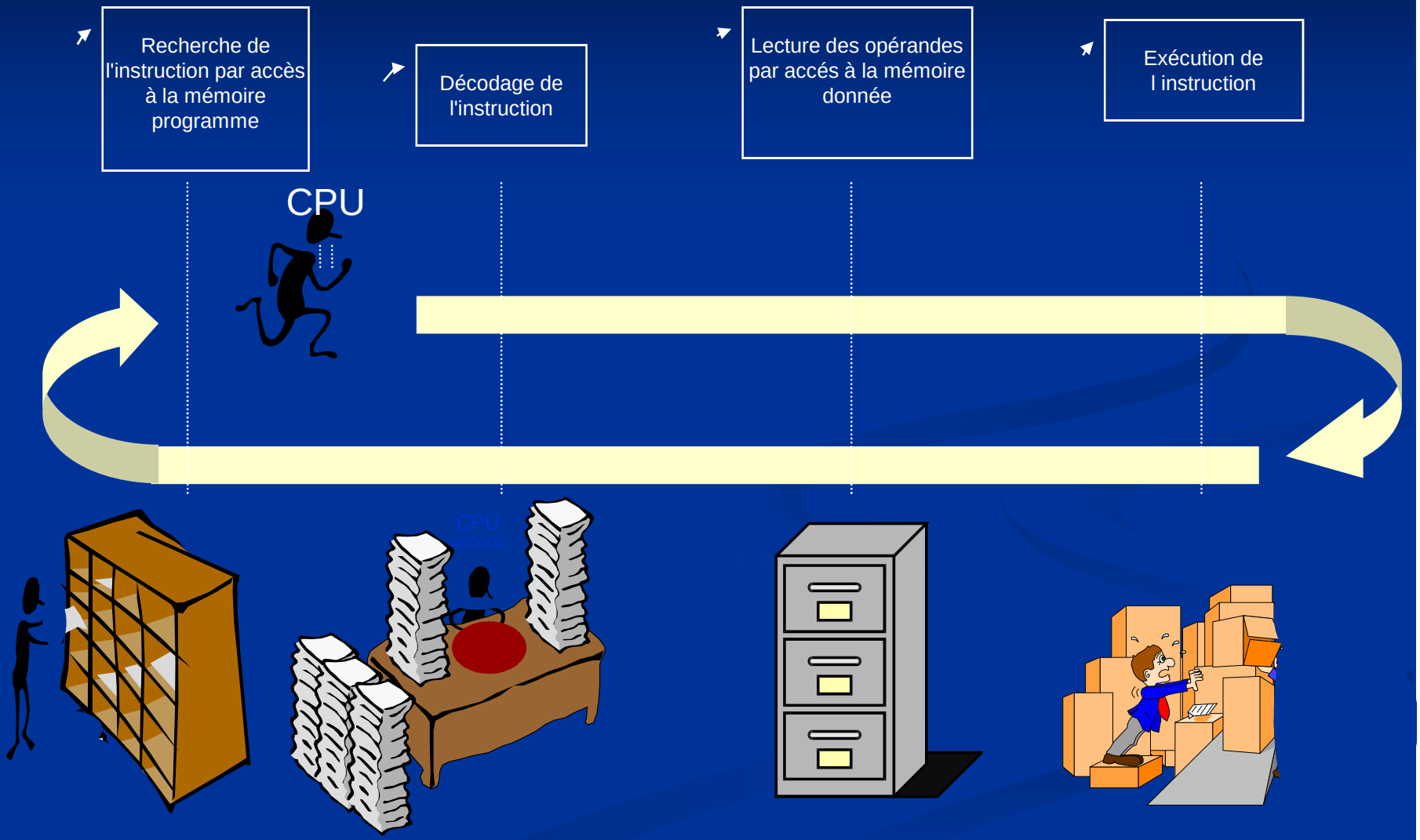
- Read Only Memory (ROM), elle conserve ses données sans énergie.



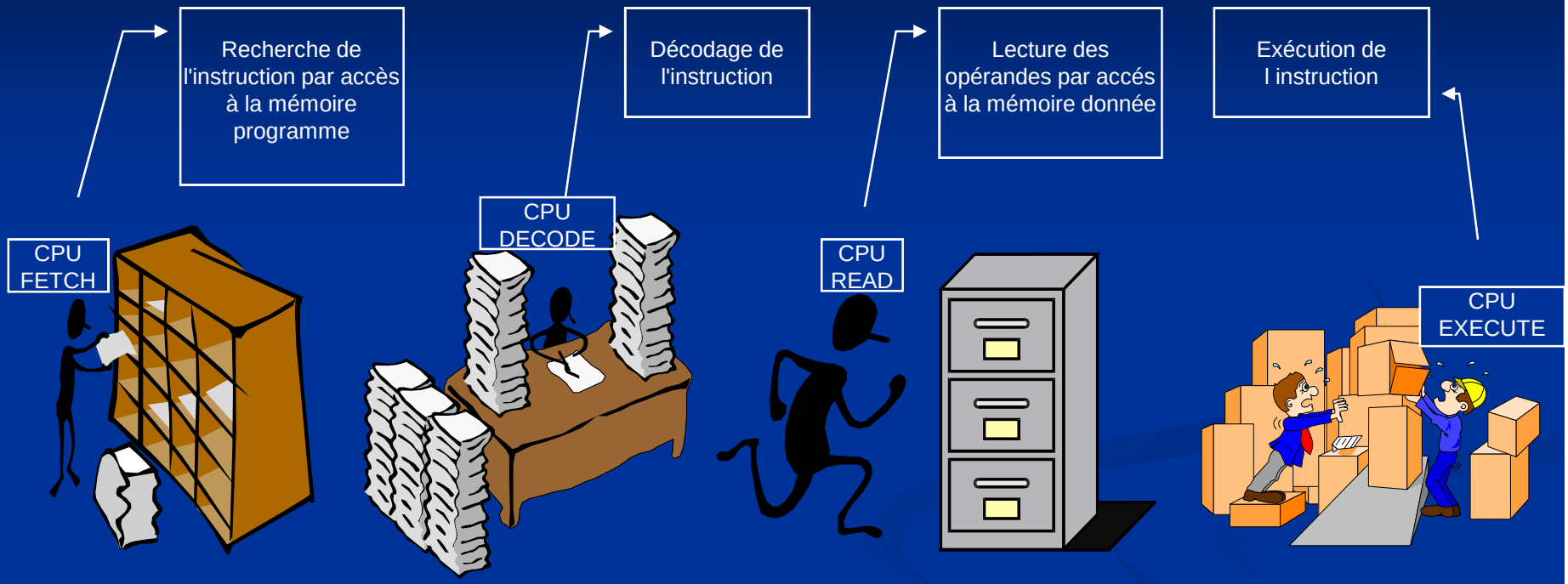
- Random Access Memory (RAM), elle ne conserve pas ses données sans énergie



Les cycles de travail du CPU



Technologie Pipeline



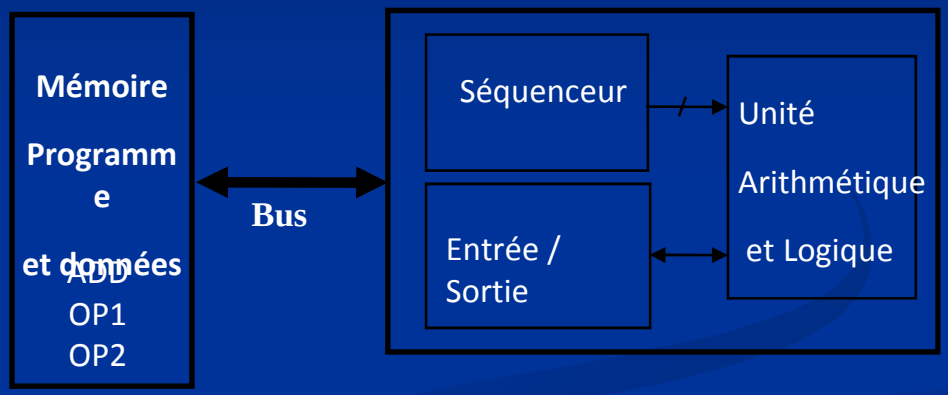
Instructions / Temps	t	t + 1	t + 2	t + 3	t + 4	t + 5	t + 6
n	F	D	R	E			
n + 1		F	D	R	E		
n + 2			F	D	R	E	
n + 3				F	D	R	E

Vitesse d 'exécution
x 4

Technologie des microcontrôleurs

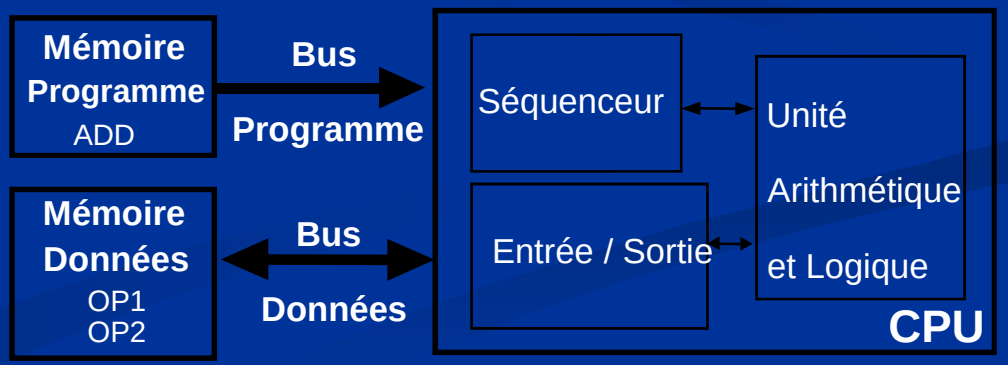
Von Neumann

John von Neumann (1903-1957), mathématicien et physicien américain d'origine hongroise



Harvard

université américaine située à Cambridge au Massachusetts.



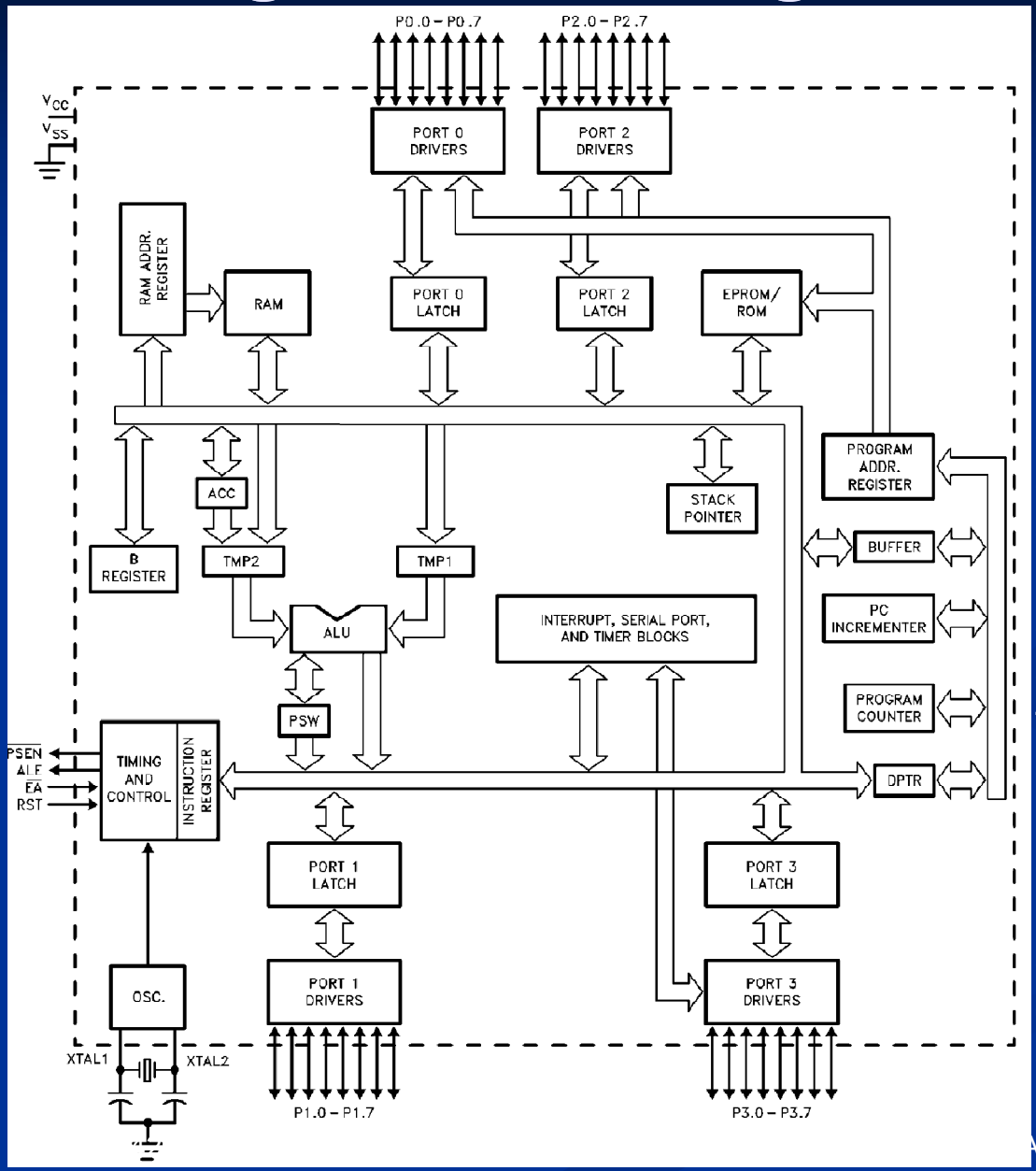
Deux concepts de jeu d'instructions pour les microcontrôleurs

- **CISC** (Complex Instruction Set Computer), 8051- 68HC11-ST6…
  - Compilateur simple
  - Architecture matérielle simple
  - Jeux d 'instructions riches
  - Au programmeur d 'optimiser le code
- **RISC** (Reduced Instruction Set Computer), PIC16, PIC18, ATMEL AVR, tous les DSP
  - Compilateur complexe
  - Architecture matérielle optimisées
  - Jeux d 'instructions succincts
  - C 'est le compilateur qui optimise

MICROCONTROLEURS – un système fermé

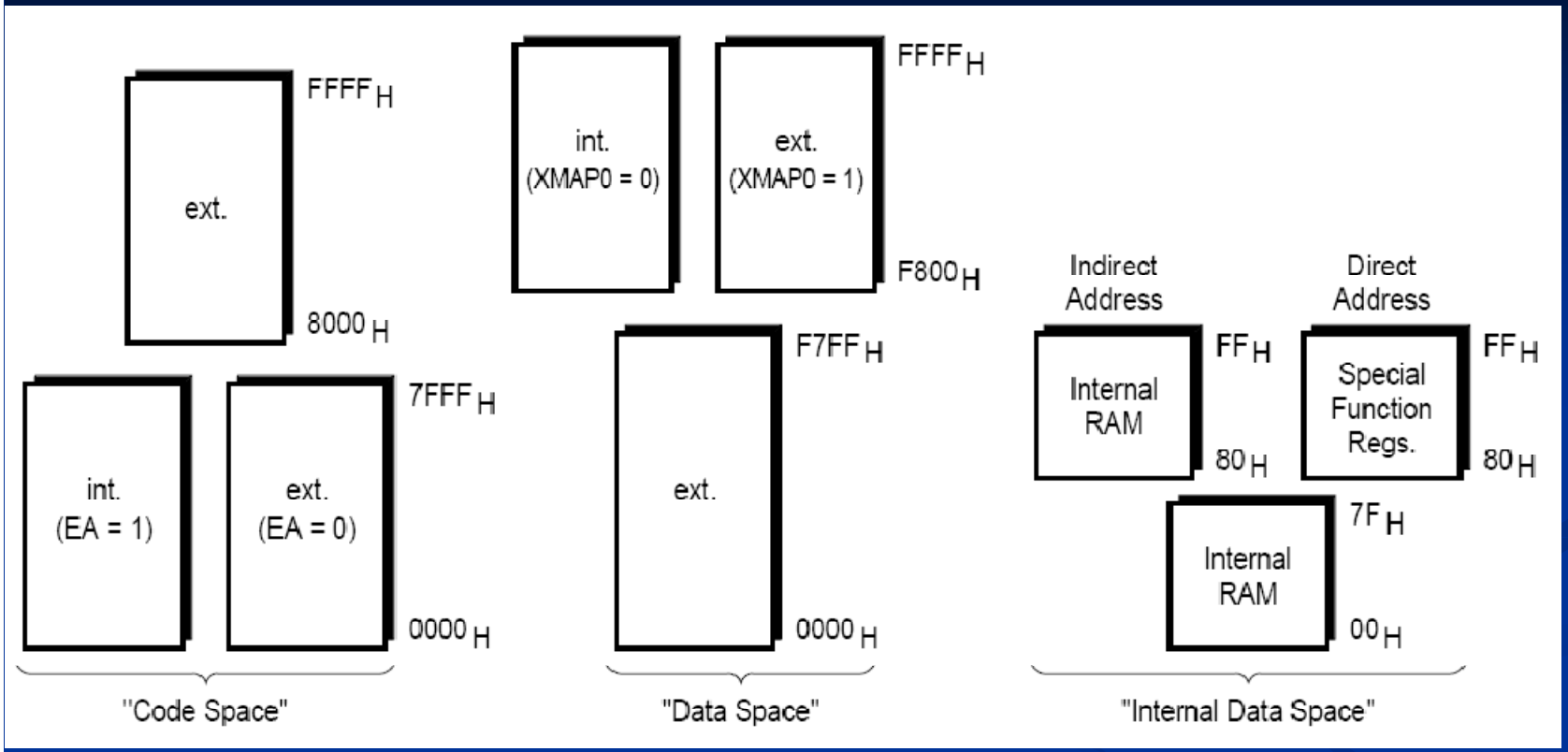
- **Avantages**
  Intégration dans un seul boîtier
  Fiabilité
  Coût de câblage réduit
  Faible consommation
- **Inconvénients**
  L'intégration de nombreux périphériques, de RAM, de ROM limite la puissance de calcul et la vitesse ces circuits.
  Mise en œuvre et approche du composant d'apparence complexe.

Organisation générale (ex:8051)



- L'ALU : Arithmetic and Logic Unit
- Les registres accumulateurs (ACC et B)
- Le registre d'état du microcontrôleur (PSW)
- Le pointeur de pile (stack pointer)
- Le gestionnaire de temps et l'oscillateur qui cadence le déroulement du programme
- Le gestionnaire d'interruptions
- Les registres pointeurs (index) DPTR
- Le compteur de programme qui contient l'adresse de l'instruction à exécuter
- Les mémoires RAM, ROM, EPROM
- Les ports parallèles, le port série, les TIMER
- La broche RST (Remise à zéro, active à l'état haut elle initialise le programme)
- ALE, Adresse Latch Enable, utilisée en mode « mémoire externe »
- /EA, si 1 la mémoire interne 0x000 à 0xFFF est sélectionnée
- /PSEN, indique un accès à la mémoire externe

Les mémoires du 8051



Mémoire programme (CODE space)
0x0000 à 0x7FFF (32KO) en interne ou 64KO en externe. *Lors d'un RESET le program counter (PC) pointe l'adresse 0x0000 (toujours le début du programme)*

Mémoire de données (DATA space) :
Une RAM externe est adressable de 0x0000 à 0xFFFF

Les adresses RAM 0x00 à 0x7F (internes) sont adressables en direct et en indirect

Plan mémoire de la RAM interne

0x00 à 0x1F : 4 banques de 8 registres généraux. (Une seule banque active) les registres R0 et R1 peuvent servir de pointeurs en RAM (adressage indirect)

.

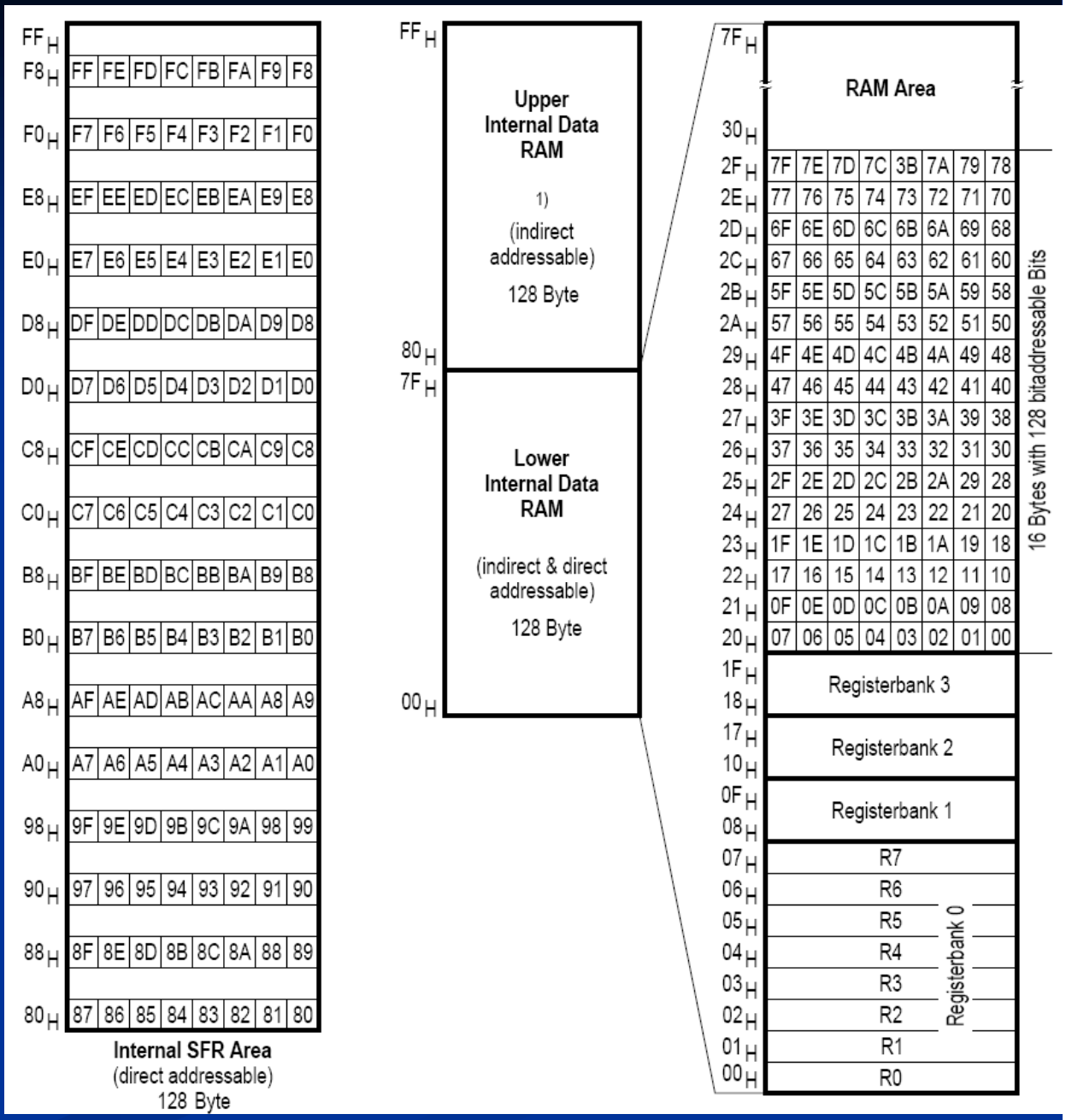
0x20 à 0x2F : 16 octets soit 128 bits adressables individuellement

0x30 à 0x7F : 80 octets

0x80 à 0xFF : 128 octets adressables uniquement en indirect

0x80 à 0xFF : 128 registres internes et de périphériques adressables uniquement en direct (les adresses finissants par 0 ou 8 sont également adressables par bit)

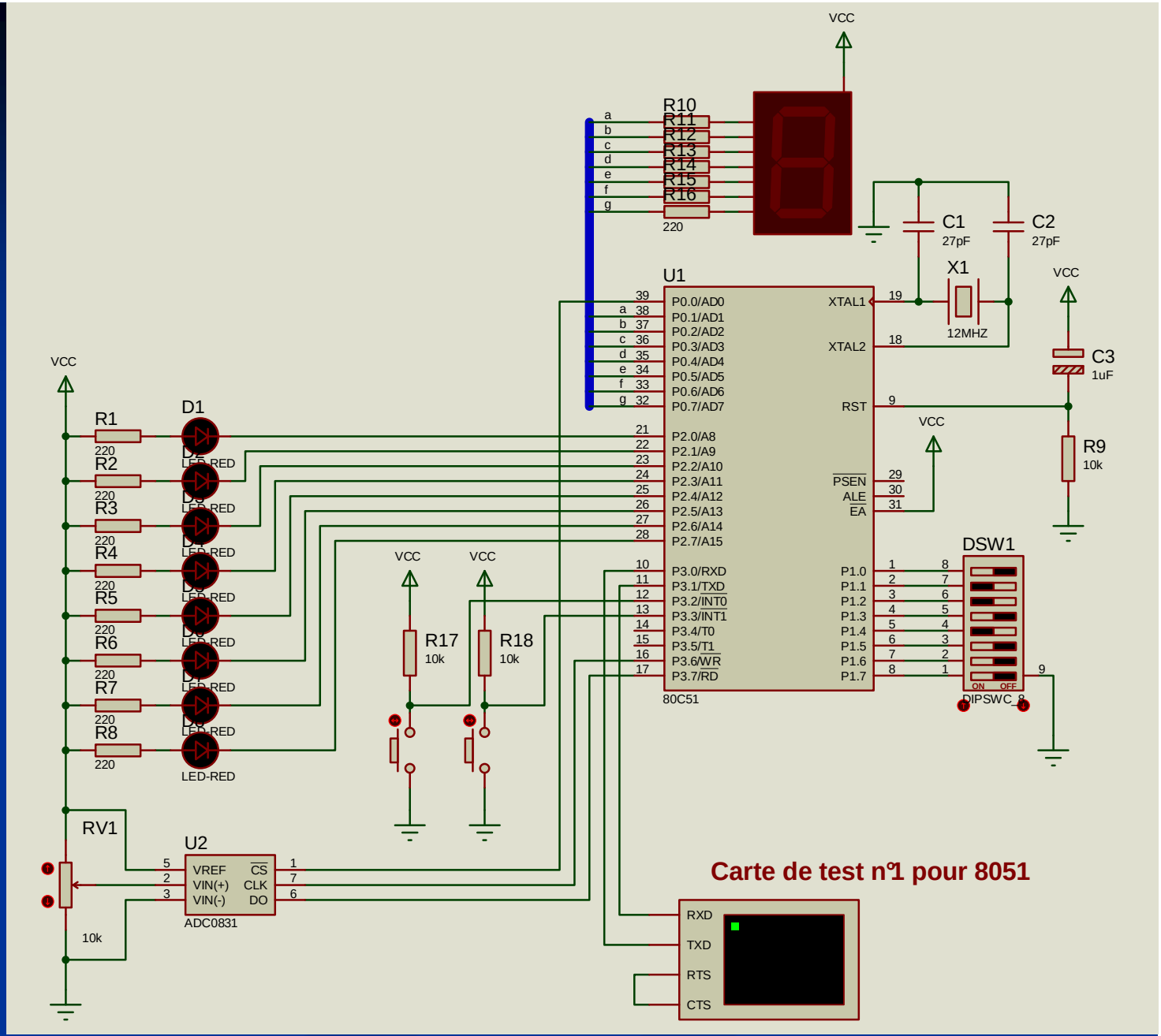
Diapo103



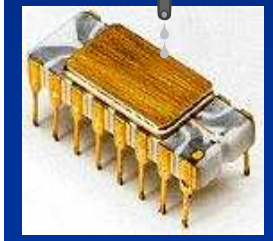
Les registres spéciaux

SYMBOL	DESCRIPTION	DIRECT ADDRESS	BIT ADDRESS, SYMBOL, OR ALTERNATIVE PORT FUNCTION MSB							LSB	RESET VALUE
ACC*	Accumulator	E0H	E7	E6	E5	E4	E3	E2	E1	E0	00H
AUXR#	Auxiliary	8EH	–	–	–	–	–	–	–	A0	xxxxxxx0B
AUXR1#	Auxiliary 1	A2H				LPEP[2]	WUPD[3]	0		DPS	xxx000x0B
B*	B register	F0H	F7	F6	F5	F4	F3	F2	F1	F0	00H
DPTR:	Data Pointer (2 bytes)										
DPH	Data Pointer High	83H									00H
DPL	Data Pointer Low	82H									00H
			AF	AE	AD	AC	AB	AA	A9	A8	
IE*	Interrupt Enable	A8H	EA	–	ET2	ES	ET1	EX1	ET0	EX0	0x000000B
			BF	BE	BD	BC	BB	BA	B9	B8	
IP*	Interrupt Priority	B8H	–	–	PT2	PS	PT1	PX1	PT0	PX0	xx000000B
			B7	B6	B5	B4	B3	B2	B1	B0	
IPH#	Interrupt Priority High	B7H	–	–	PT2H	PSH	PT1H	PX1H	PT0H	PX0H	xx000000B
			87	86	85	84	83	82	81	80	
P0*	Port 0	80H	AD7	AD6	AD5	AD4	AD3	AD2	AD1	AD0	FFH
			97	96	95	94	93	92	91	90	
P1*	Port 1	90H	–	–	–	–	–	–	T2EX	T2	FFH
			A7	A6	A5	A4	A3	A2	A1	A0	
P2*	Port 2	A0H	AD15	AD14	AD13	AD12	AD11	AD10	AD9	AD8	FFH
			B7	B6	B5	B4	B3	B2	B1	B0	
P3*	Port 3	B0H	$\overline{RD}$	$\overline{WR}$	T1	T0	$\overline{INT1}$	$\overline{INT0}$	TxD	RxD	FFH
PCON#[1]	Power Control	87H	SMOD1	SMOD0	–	POF	GF1	GF0	PD	IDL	00xx0000B
			D7	D6	D5	D4	D3	D2	D1	D0	
PSW*	Program Status Word	D0H	CY	AC	F0	RS1	RS0	OV	–	P	000000x0B
RCAP2H#	Timer 2 Capture High	CBH									00H
RCAP2L#	Timer 2 Capture Low	CAH									00H
SADDR#	Slave Address	A9H									00H
SADEN#	Slave Address Mask	B9H									00H
SBUF	Serial Data Buffer	99H									xxxxxxxxB
			9F	9E	9D	9C	9B	9A	99	98	
SCON*	Serial Control	98H	SM0/FE	SM1	SM2	REN	TB8	RB8	TI	RI	00H
SP	Stack Pointer	81H									07H
			8F	8E	8D	8C	8B	8A	89	88	
TCON*	Timer Control	88H	TF1	TR1	TF0	TR0	IE1	IT1	IE0	IT0	00H
			CF	CE	CD	CC	CB	CA	C9	C8	
T2CON*	Timer 2 Control	C8H	TF2	EXF2	RCLK	TCLK	EXEN2	TR2	C/$\overline{T2}$	CP/$\overline{RL2}$	00H
T2MOD#	Timer 2 Mode Control	C9H	–	–	–	–	–	–	T2OE	DCEN	xxxxxx00B
TH0	Timer High 0	8CH									00H
TH1	Timer High 1	8DH									00H
TH2#	Timer High 2	CDH									00H
TL0	Timer Low 0	8AH									00H
TL1	Timer Low 1	8BH									00H
TL2#	Timer Low 2	CCH									00H
TMOD	Timer Mode	89H	GATE	C/$\overline{T}$	M1	M0	GATE	C/$\overline{T}$	M1	M0	00H

Mise en œuvre, exemple du 8051



Le langage ASSEMBLEUR



Langage Machine

```
0x8008   29
0x8009   7F 20
0x800B   B4 0400
```

Adresse — Code opération — Opérandes

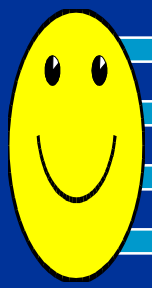
Assemblage

Langage Assembleur

```
LOOP:  ADD    A, R1
       MOV    R7,# 0x20
       CJNE   A,#0x04, LOOP
```

Label — Code opération — Opérandes

- **Avantages / Inconvénients**



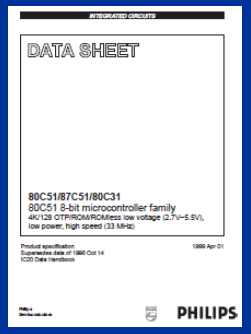
- Code généré très court
- Demande moins de mémoire
- Exécution très rapide
- Outils de développement simple

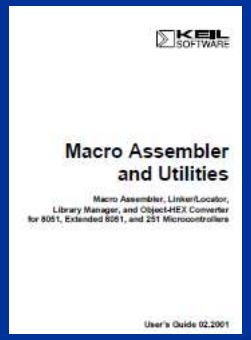
- Près du code machine
- Long à écrire
- Difficile à lire
- Difficile à mettre au point
- Dépendant du micro

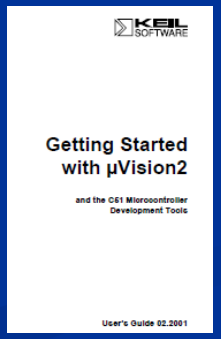
- **Conclusions**
  - Routines temps réel en assembleur
  - Corps principal en C
  - Importance de la documentation

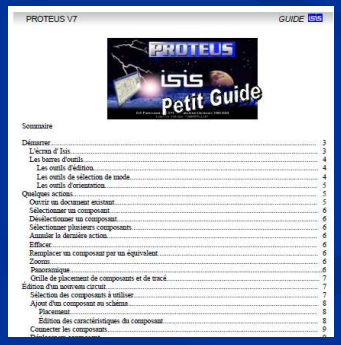
Pour programmer un microcontrôleur…

- Il faut **se documenter** :
  - Connaitre le composant, son unité centrale, ses périphériques etc.
  - Connaitre l'assembleur, le jeu d'instruction, les directives d'assemblage
  - Savoir mettre en œuvre les outils de développement, simulateur, émulateur, programmateur etc.









Elément d'assembleur 8051

- ACC est l'accumulateur. Il comprend 8 bits.

```
mov    a,# 12          ; charge la valeur 12 dans le registre ACC
```

- B est utilisé pour les opérations de multiplication et de division

```
mul    AB ;AxB , résultats pFORTS dans B, pfaibles dans ACC
div    AB ;A/B, résultat dans ACC, reste dans B
```

- Les registres R0 et R1 sont des registres pointeurs pour la mémoire RAM

```
mov    a,70h       ;Charge 70h dans ACC
mov    71h,a       ;enregistre ACC dans la case 71h
```

- Au lieu de cette séquence on peut écrire

```
mov    R0,70h          ;R0 pointe l'adresse 70h
mov    a,@R0           ;le contenu de 70h est placé dans ACC
inc    R0      ;R0 pointe l'adresse 71h
mov    @R0,a       ;le contenu de ACC est placé à l'adresse 71h
```

Modes d'adressage, 4 modes pour une même instruction

- `ADD A,# 127` (adressage immédiat)
  A=A+127
- `ADD A,7FH` (adressage direct)
  A=A+contenu adresse 7F
- `ADD A,R7` (adressage registre)
  A=A+contenu de R7
- `ADD A,@R0` (adressage indirect)
  A=A+contenu de l'adresse pointé par R0

Modes d’adressage, accès ROM

- **Base register plus index register addressing :** L’adresse de la donnée est calculée en ajoutant le décalage ACC au registre DPTR (data pointer) ou PC.
-

```
clr    a          ; ACC=0
mov    DPTR,#0A34h  ; DPTR=A34h
movc   a,@a+DPTR  ; ACC égale le contenu de la mémoire ROM A34h
```

Le jeu d'instructions- Syntaxe

- Rn – registres R0-R7
- `@Ri` – adressage indirect RAM interne ou externe par les registres R0 ou R1
- `#data 8` – adressage immediat (8 ou 16 bits)
- `bit` – 128 drapeaux généraux et bits des SFR
- A – Accumulateur ACC

Les types d'instructions

- Logiques `andl a,#12 ; a=a&12`
- Arithmétiques `add a,#12 ; a=a+12`
- Transfert `mov a,#12 ; a=12`
- Booleen `setb #12 ; bit 12 = 1`
- Contrôle du programme : `jb P3.0,ici ; saut à l'adresse « ici » si P3.0=1`

Exemple assembleur

Quatre « champs » : étiquettes (adresses), instructions, opérandes, Commentaires …)

```
#include <reg517A.inc>
; CD 6/08
; PROGRAMME FLASH LED PORT20
; ce programme fait clignoter (flash) le port P2.0
;
LED EQU    P2.0    ; utilisation de la LED sur P2.0
    CSEG   AT 0    ; adresse d'assemblage (ici 0h, RESET)
    MOV SP,#7Fh ; charge SP avec sommet de la RAM
SUITE:  CPL LED ; complémente la LED (FLASH)
    CALL   TEMPO       ; tempo
    SJMP   SUITE       ; on recommence
; TEMPO LONGUE R0xR1
TEMPO:  MOV R0,#0FFh   ; charge R0 et R1
TEMPO1: MOV R1,#0FFh
TEMPO2: DJNZ   R1,TEMPO2 ; decremente R1 jusqu'à 0
    DJNZ   R0,TEMPO1 ; decremente R0 et recharge R1 si R0!=0
    RET
    END
```

OUTILS DE DEVELOPPEMENT

Outils de développement

```
#include
<stdio.h>
main()
{ puts(" Bonjour à
tous ");
}
```

Fichier source C contenant la fonction main

Bibliothèques en C (texte) *.c

Fichiers d'inclusion *.h

Bibliothèques pré-compilées (fichiers objet)

Préprocesseur
Met en forme le fichier source C à partir de **directives** #define, #include

Compilateur C
Transforme le fichier C en un fichier objet (code machine), les fonctions pré-compilés sont déclarées dans les fichiers *.h

Fichiers d'inclusion *.inc

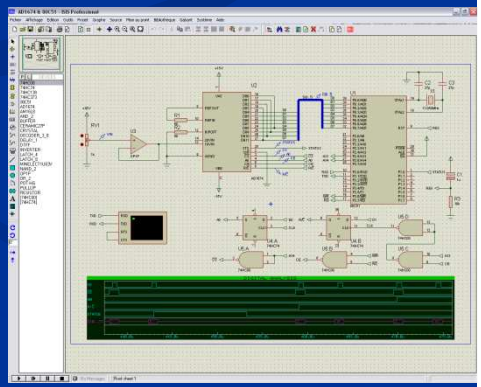
Editeur de lien **LINKER**
Lie (donne des adresses aux fonctions) tous les fichiers objets et crée un fichier exécutable

Programmes en assembleur *.a

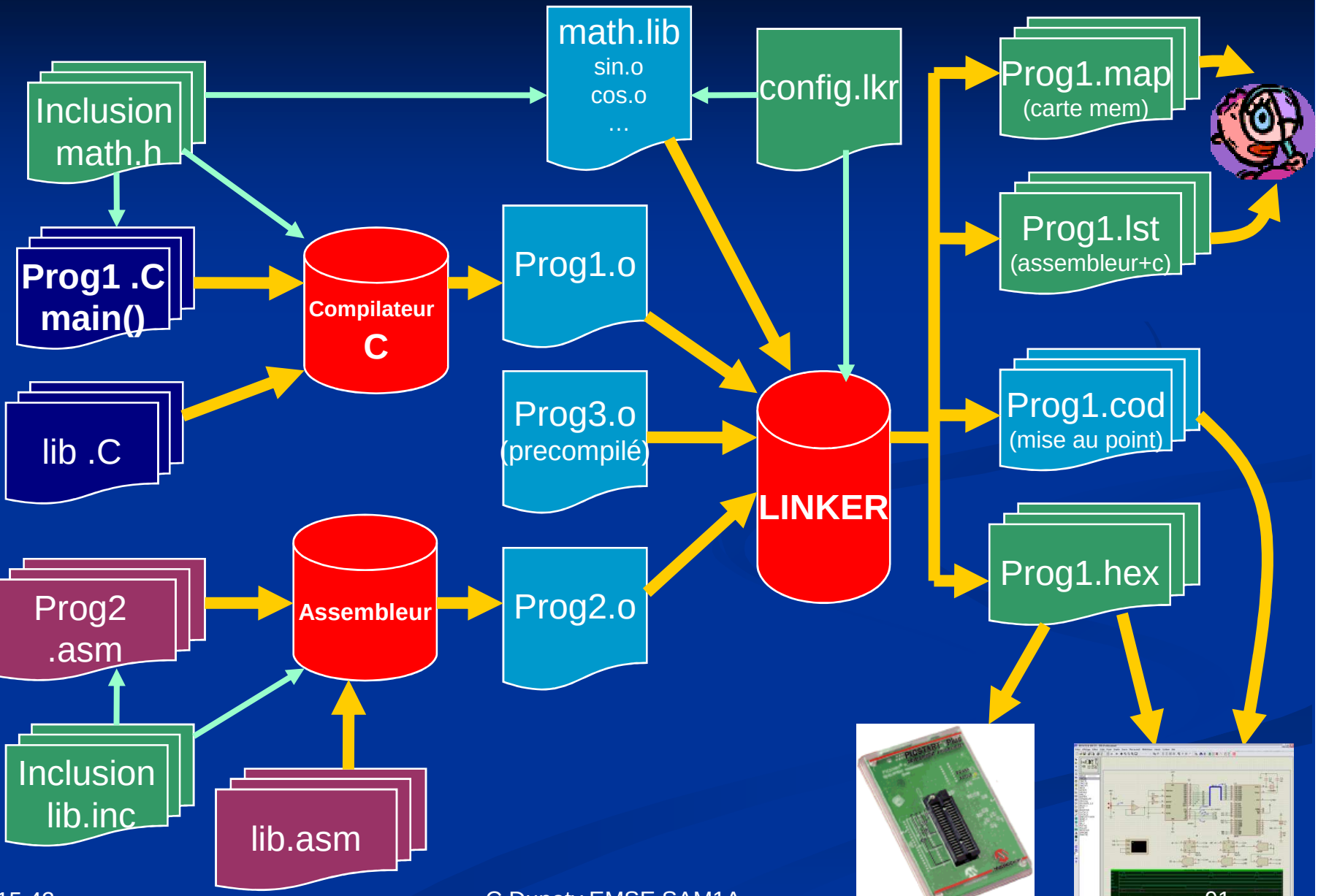
ASSEMBLEUR
Transforme le fichier .a en un fichier objet (code machine)

Fichiers pour debug
Prog.cod
Prog.lst

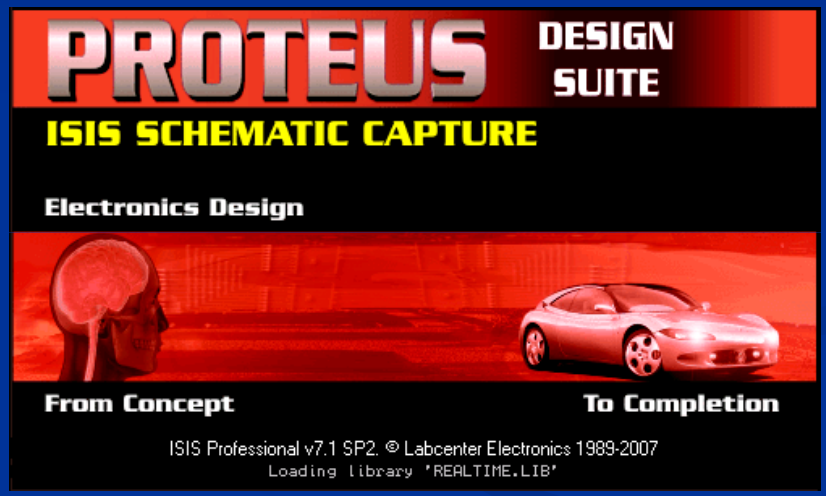
Programme exécutable
Prog.hex
(+ Prog.err)



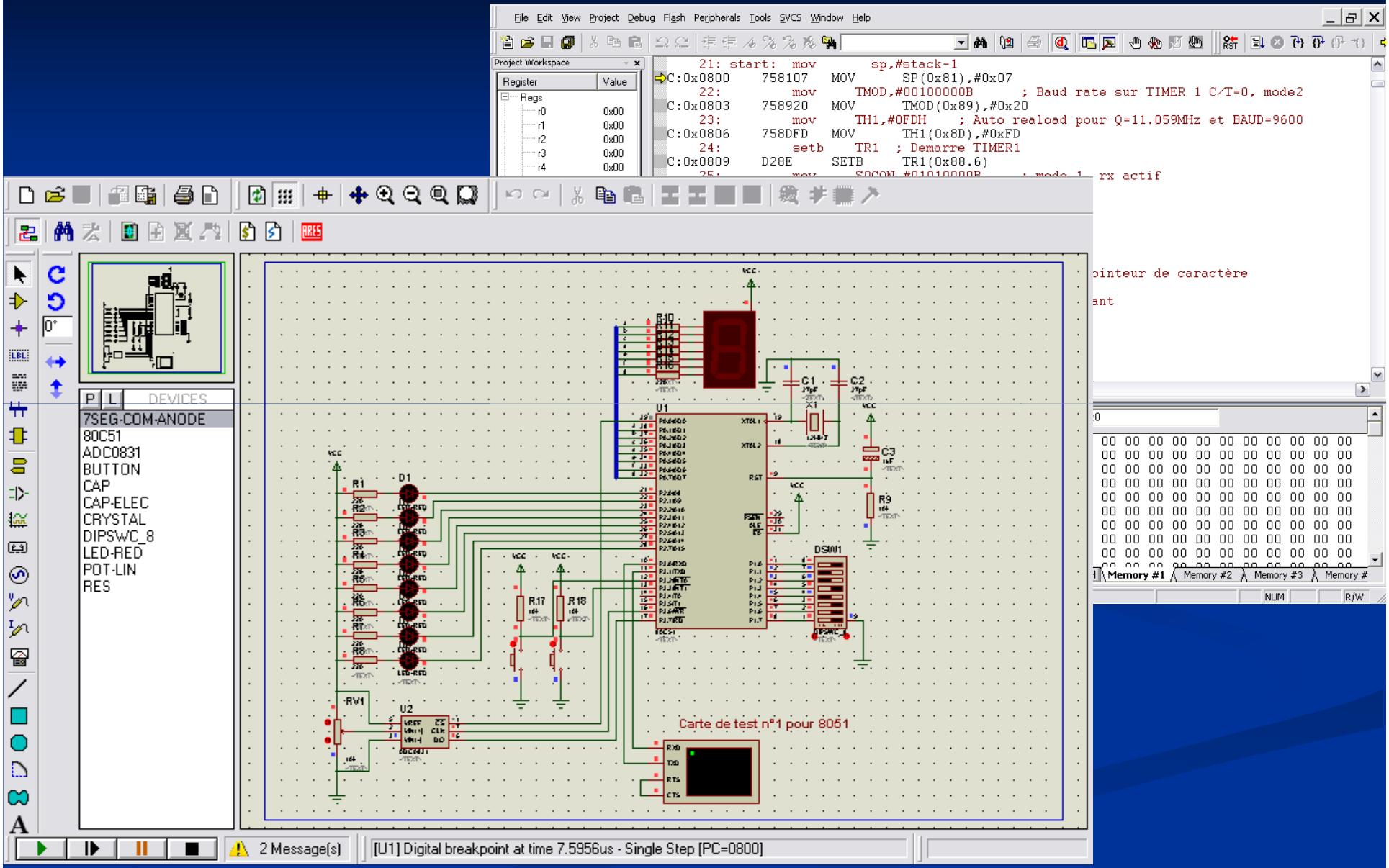
Flux des données



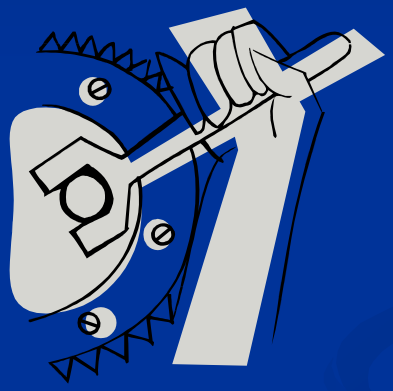
uVISION et PROTEUS-ISIS-VSM



Simulation-exemple



TP N°1



Les modes d’adressages, Précisions et compléments

MODE D 'ADRESSAGE (1)

Immédiat

MOV A, # 0x7F

Base	Suffix	Legal Characters	Examples
Hexadecimal	H, h	0 – 9, A – F, a – f	0x1234 0x99 1234H 0A0F0h 0FFh
Decimal	D, d	0 – 9	1234 65590d 20d 123
Octal	O, o, Q, q	0 – 7	177o 25q 123o 177777q
Binary	B, b	0 and 1	10011111b 101010101b

Base numérique

Symbole de l'adressage immédiat

A ?

Exemples:

Valeur numérique

```
MY_VAL      EQU     50H
```

Directive assembleur

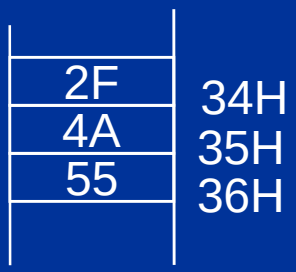
```
MOV         A,#0E0h
MOV         DPTR,#0x8000
ANL         A,#128
XRL         A,#0FFh
MOV         R5,#MY_VAL
```

Assembleur

Direct

MOV A, 0x35 ? A

Adresse Effective = la valeur
Adresse possible 0 .. 255
Permet l 'accès à la RAM basse et aux SFRs



Exemples:

```
VALUE:      DS      1
IO_PORT2    DATA    0A0H
VALUE2      DATA    20H
```

Directives assembleurs

Adresse mémoire: 0 .. 255

```
MOV         A,IO_PORT2
ADD         A,VALUE
MOV         VALUE2,A
MOV         R1,#VALUE
```

Assembleur

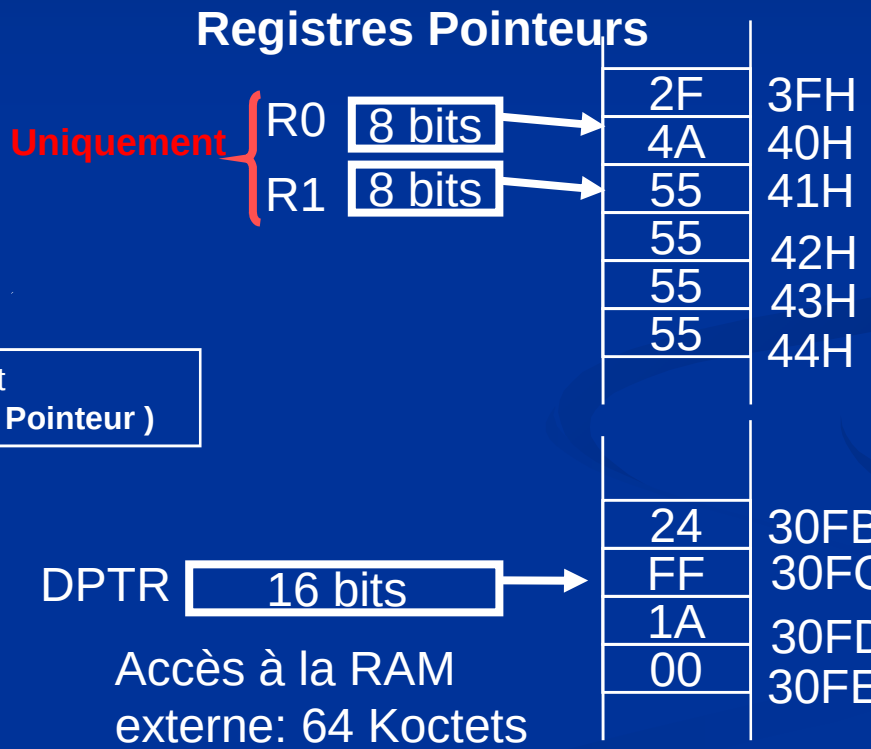
- **Direct par registre**

  MOV R3, A

- **Indirect par registre**

```
MOV     A,  # 0x56
MOV     0x40, A
MOV     R0, # 0x40
MOV     0x43, @R0
```

Symbole de l 'adressage indirect
Adresse Effective = (Registre Pointeur)



Exemples:

```
BUFFER:     DS      100
MOV         R0,#BUFFER
MOV         A,@R0
INC         R0
MOV         @R0,A
```

```
XBUFFER:  DS   30FCh
VAR1:     DS   1
MOV       DPTR,#XBUFFER
MOVX      A,@DPTR
MOV       R1,#VAR1
MOVX      @R1,A
```

Indirect indexé par registre

MOVC A, @ A+PC
MOVC A, @ A+DPTR

Adresse Effective = (DPTR ou PC) + (A)
= (Base Adresse) + (Index)

Exemples:

```
TABLE:      DB      1,2,4,8,0x10
MOV         DPTR,#TABLE
MOV         A,#3
MOVC        A,@A+DPTR
```

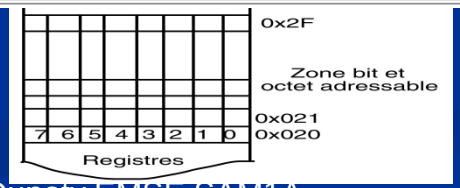
Bit adressage

Bit désigné par son poids de 0 à 7

UAL de bit

ANL C, ACC.7
MOV C, 0x21.3

0xF8	P5							
0xF0	B							
0xE8	P4							
0xE0	ACC							
0xD8	ADCON	ADDAT	DAPR					
0xD0	PSW							
0xC8	T2CON		CRCL	CRCH	TL2	TH2		
0xC0	IRCON	CCEN	CCL1	CCH1	CCL2	CCH2	CCL3	CCH3
0xB8	IEN1	IP1						
0xB0	P3							
0xA8	IEN0	IP0						
0xA0	P2							
0x98	SCON	SBUF						
0x90	P1							
0x88	TCON	TMOD	TL0	TL1	TH0	TH1		
0x80	P0	SP	DPL	DPH				PCON



Zone mémoire +
SFRs bits adressables

Exemples:

```
FLAG:       DBIT    1
P1          DATA    90H
GREEN_LED   BIT     P1.2
```

```
            SETB    GREEN_LED
            JB      FLAG,is_on
            SETB    FLAG
            CLR     ACC.5
             :
is_on:      CLR     FLAG
            CLR     GREEN_LED
```

Le registre d'état : Program Status Word (PSW)

BIT 7	BIT 6	BIT 5	BIT 4	BIT 3	BIT 2	BIT 1	BIT 0
CY	AC	F0	RS1	RS0	OV	UD	P

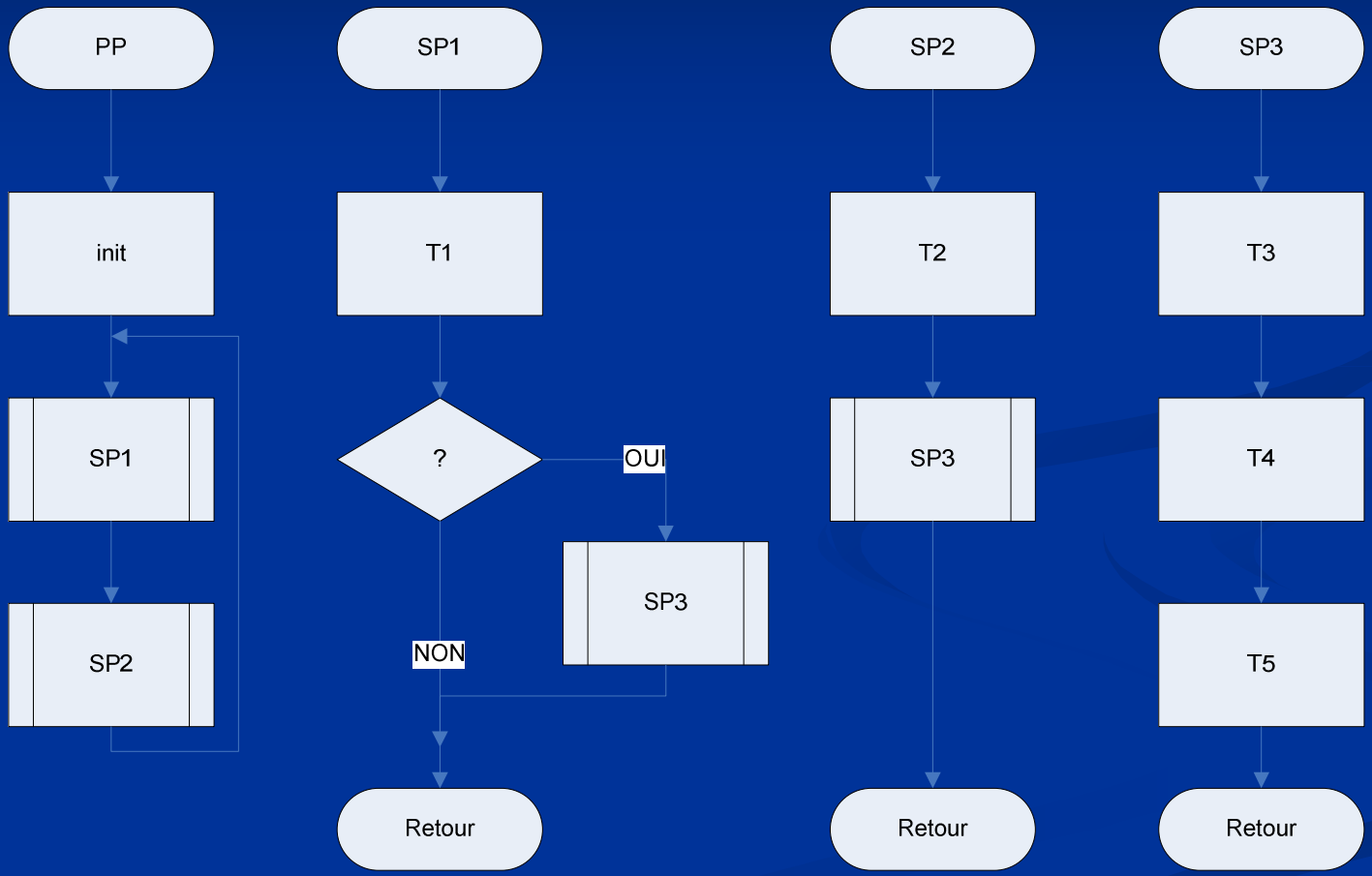
CY Carry flag, indique une retenue
AC Auxiliary Carry flag (pour les opérations BCD)
F0 Flag 0 (un indicateur à disposition de l'utilisateur)
RS1, RS0
Register bank select: RS1 RS0 Working Register Bank and Address
0 0 Bank0 (D:0x00 - D:0x07)
0 1 Bank1 (D:0x08 - D:0x0F)
1 0 Bank2 (D:0x10 - D:0x17)
1 1 Bank3 (D:0x18H - D:0x1F)
OV drapeau de débordement (Overflow flag)
UD User definable flag
P drapeau de parité (Parity flag)

```
jb  C,labas
jnb F0,encore
```

Les directives d'assemblage

- Commentaire : ;
- Etiquette : `ici:` …
- Equivalence : `EQU var1 EQU 123`
- Réservation en RAM : `DS label : DS 5`
- Initialisation de constantes : `DB ici : DB 27,33h,'coucou'`
- Initialisation de const 16bits : `DW la : DW 123 ,ABCDh`
- Réservation d'un bit `DBIT flag : DBIT 2`
- Adresse courante `$ jmp $`
- `Fin du texte assembleur END`

Programmation structurée



Sous-programmes, la pile S

L'instruction call est codée sur deux octets
call tempo se trouve à l'adresse 0x123
call calcul se trouve à l'adresse 0x456

```
mov     s,#7Fh
call    tempo
…
jmp     ailleurs

tempo: …
call    calcul
…
ret

calcul:…
…
ret
```

```
call    calcul

call    tempo

mov     s,#7Fh
```

AD RAM pointée par SP	Valeur
0x84	?
0x83	58h
0x82	04h
0x81	25h
0x80	01h
0x7F	?

ret

ret

La définition des segments mémoires

- DATA : mémoire RAM et SFR de 0x00 à 0xFF, adressable en direct et indirect
- BIT : zone RAM « bit adressable » et certains SFR, adressables avec les instructions pour bits.
- IDATA : RAM adressable en indirect par les registres R0,R1, (généralement 0x00 à 0xFF)
- XDATA : RAM externe accessible par l'instruction MOVX via le registre DPTR.
- CODE : ROM accessible via MOVC et le registre DPTR

- Nommer un segment : SEGMENT
- Sélectionner un segment : RSEG
- Segment pré-definis : CSEG (ROM), DSEG (DATA), BSEG(BIT), ISEG (IDATA, XSEG (XDATA)

- La directive ORG permet de changer d'adresse DANS un segment

Structure mémoires

Utilisation du Linker La définition des segments

```
mesdonnees      SEGMENT DATA   ; segment de DATA
monprog         SEGMENT CODE   ; ROM
mesconst        SEGMENT CODE   ; ROM
pile            SEGMENT IDATA  ; place pour la pile en IDATA

        RSEG    mesdonnees
val :   DS      1              ; réserve un octet en RAM DATA

        RSEG    pile           ; reservation pour pile en IDATA
        DS      20h

        CSEG    AT 0           ; autre methode
        LJMP    debut          ; saut sur debut
                …
```

Suite 

```
        RSEG    monprog         ; zone ROM connue du linker
debut: MOV      SP,#pile-1
        mov     a,#0ABh
        mov     val,a
…
        RSEG    mesconst        ; une autre zone ROM
Const: DB       27,33h,'coucou',0

        RSEG    mesdonnees      ; on se place en RAM
Donnee :      DS      20              ; reserve 20 octets en RAM

        RSEG    monprog       ; suite du programme en ROM
        inc     a
        mov     R0,val
        mov     @R0,a
        jmp     $
```

QUIZZ

- A quoi sert l'accumulateur, le Program Counter ?
- Le PC pointe toujours sur une instruction ?
- Dans quel registre se trouve les Flags liés à l 'A.L.U ?
- Pour obtenir une capacité d 'adressage de 128 Koctets combien de lignes d 'adresse faut-il ?
- Une Pile LIFO peut se placer n 'importe ou dans l 'espace mémoire du microcontrôleur.
- C 'est au programmeur de gérer les déplacements pointeur de pile.
- En quoi un débordement de pile est catastrophique.
- Le bus d 'adresses permet d 'accéder à la fois aux constantes et au code, mais pas aux périphériques.
- Quels sont les avantages de la programmation structurée
- Un registre est une case mémoire accessible comme une autre
- Pourquoi dans les systèmes embarqués on utilise des RAMs, des EPROMs et pas des disques durs.
- Intérêt des mémoires FLASH ?, remplacent-t-elle les EEPROMs ?
- Dans un prcesseur CISC un cycle d 'instruction machine correspondant-il toujours à une période d 'horloge ?

QUIZZ

- Différence entre MOV, MOVC ?

- Quel est l 'état des registres R0, R1, A, B, du port 1 et de la case mémoire 40H , après l 'exécution du programme suivant ?

```
MOV     R0, #30H
MOV     A,  @R0
MOV     R1, A
MOV     B, @R1
MOV     @R1, P1
MOV     P2, P1
```

Donnees	Adresses
0CAH	P1
40 H	30H
...	
10H	40H

- Quel est l 'état de la carry, du port 1 et de la case mémoire 30H après l 'exécution du programme suivant ?

```
SETB    C
MOV     P1.3, C
MOV     C, P3.3
MOV     P1.2, C
CLR     C
MOV     30H.6, C
```

11000101B	P3
35H	P1
40 H	30H
...	

- Il y a une erreur dans le programme précédent, laquelle ?

- Quelles sont les valeurs des registres DPH et DPL après l 'instruction suivante ?

```
MOV     DPTR, #4660
```

- Quel est l 'état des cases mémoires ci-dessous après l 'exécution du programme suivant ?

Donnees	Adresses
40 H	1FFFH
...	
0CDH	2FA0H

Mémoire Programme

Donnees	Adresses
1F H	1FFFH
...	
00H	2FA0H

Mémoire Donnée

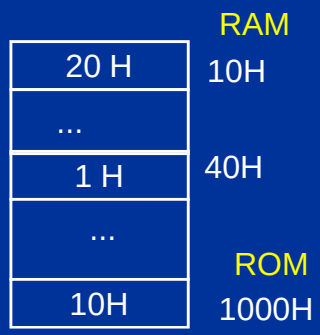
```
Initialisation routine
MOV DPSEL, #06H
MOV DPTR, #1FFFH
MOV DPSEL, #07H
MOV DPTR, #2FA0H

Table transfert routine
CLR  A
MOV DPSEL, #06H
MOVC      A, @A + DPTR
MOV DPSEL, #07H
MOVX      @DPTR, A
```

❒ Corriger les erreurs

```
MOV     DPTR,#0x1FFF0
MOVC    A,@DPTR
MOV     DPTR, #0x10  ;pointeur en RAM interne
MOV     A,@A+DPTR
MOV     R0,#0x1000
MOV     DPTR,#FA20H
```

❒ Quelle est la valeur contenu dans l 'accumulateur à la fin du programme suivant:

```
MOV     DPTR,#1000H
MOVC    A,@A+DPTR
MOV     R0, A
MOV     A, #2
ADD     A, R0
ADD     A, #2
ADD     A,@R0
ADD     A, 40H
```



❒ Quelle est la valeur contenu dans l 'accumulateur à la fin du programme suivant ?


```
MOV     R1, #0DBH
MOV     A, 0A0H
ADD     A,@R1
```

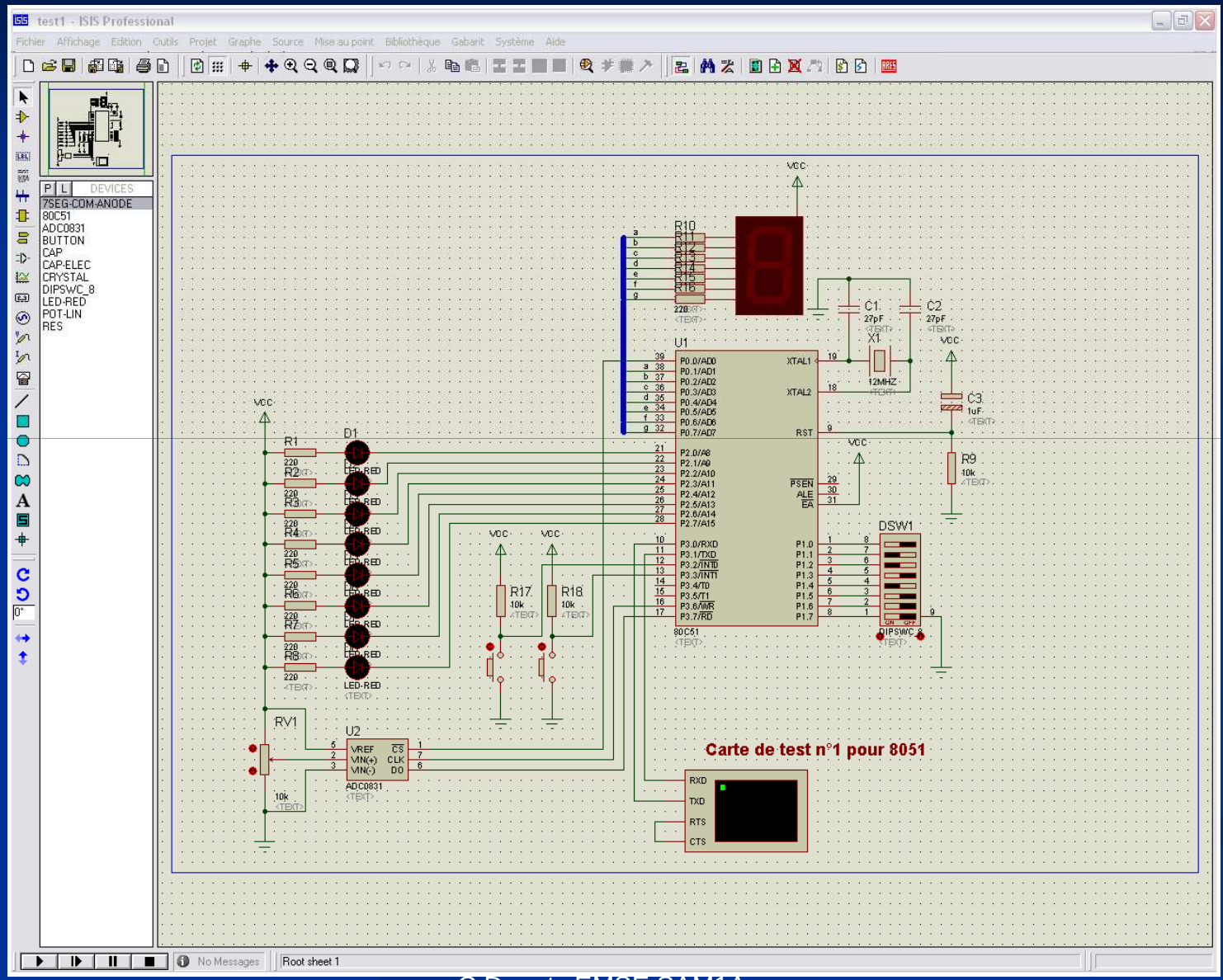
❒ Qu 'appelle t-on vecteur de reset ?

❒ Quelle est la directive assembleur qui permet de placer le code sur le vecteur de RESET?

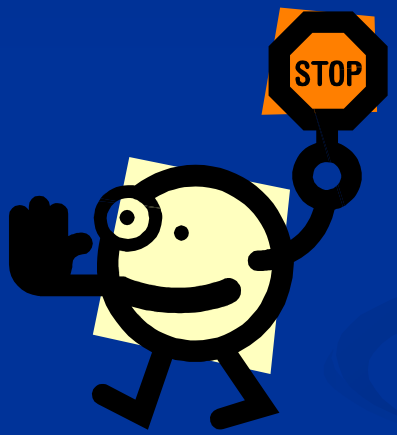
TP N°2

Utiliser les outils de développement
Découvrir le jeu d'instruction
Créer et tester des programmes simples

ISIS VSN & uVISION

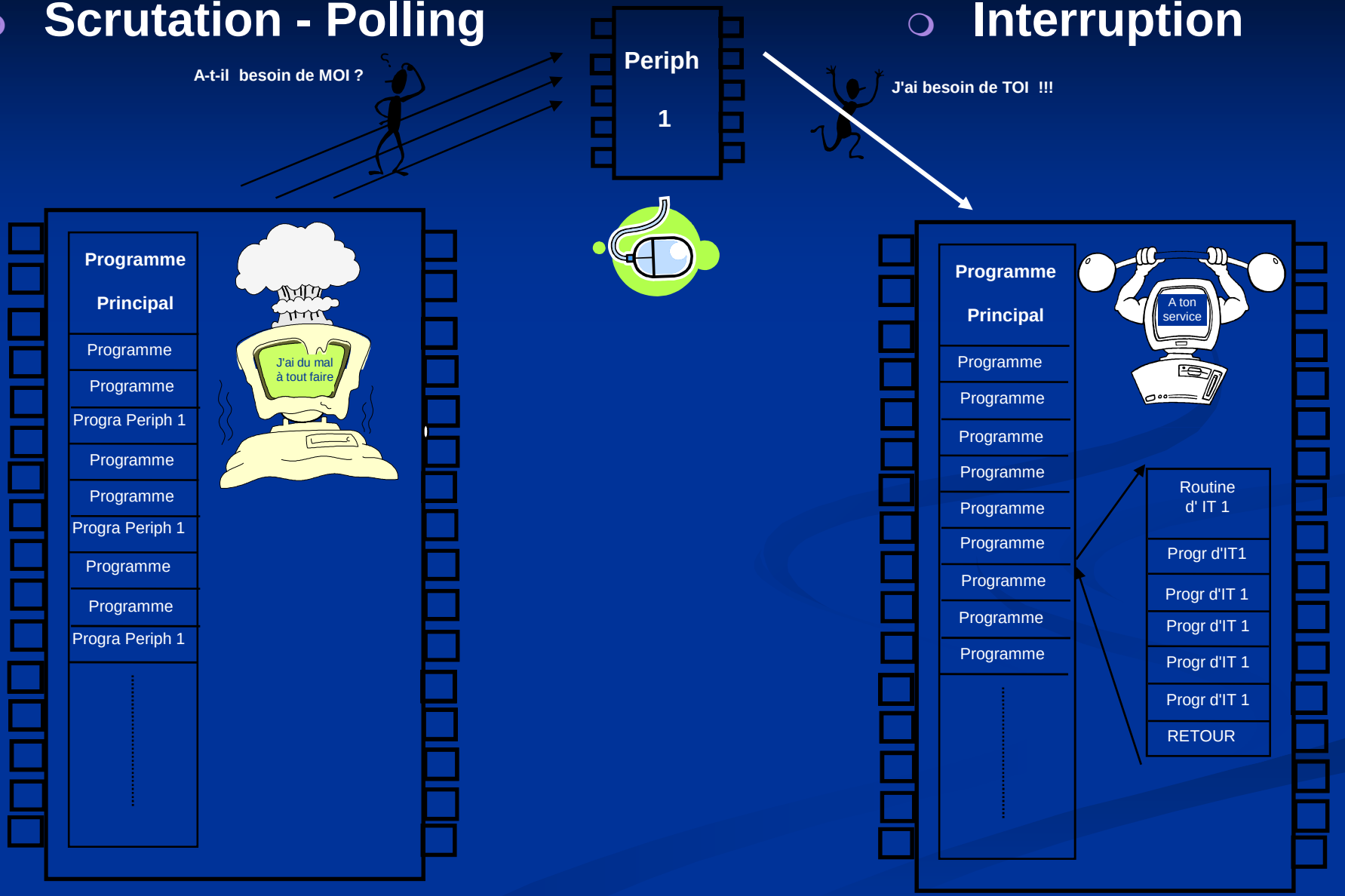


Les interruptions

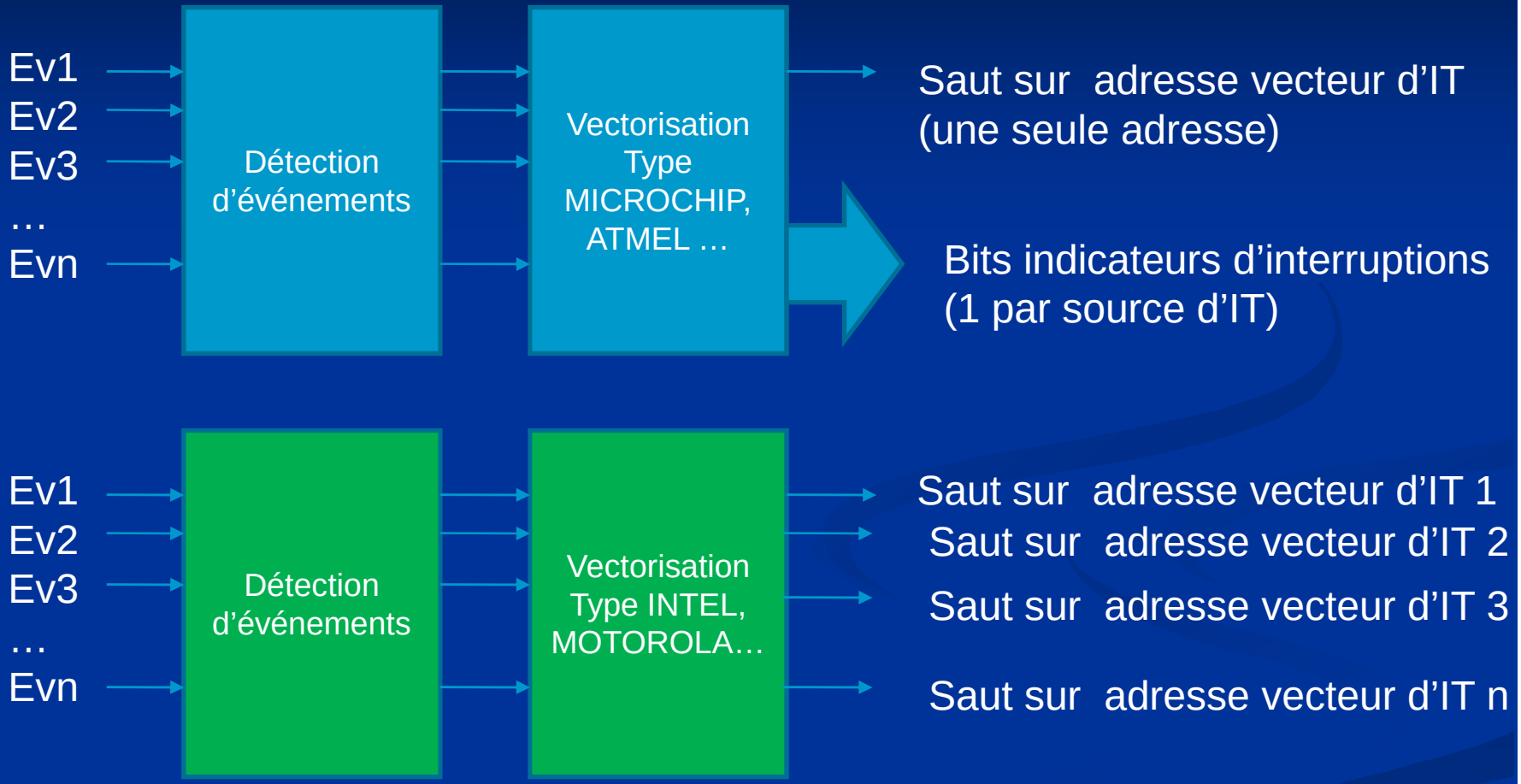


Scrutation VS Interruptions

- **Scrutation - Polling**
- **Interruption**



Deux approches



interruptions

- **Validation globale**

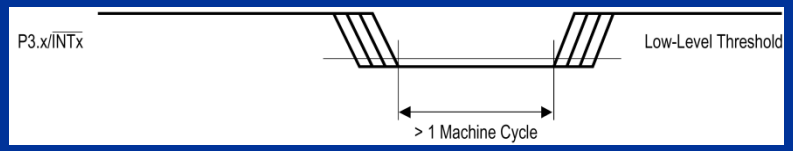
  Inhibition ou Validation de toutes les interruptions

- **Niveau de priorité**

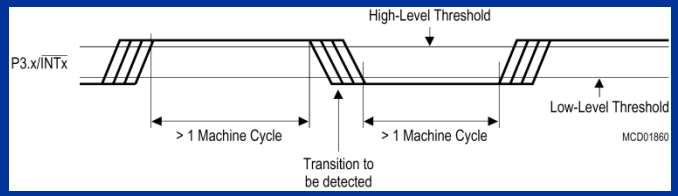
  Interruptible si priorité N+1

- **Type de déclenchement**

  - **Sur niveau**



  - **Sur front**



- **Routine d 'interruption**

  Sous-programme d 'IT

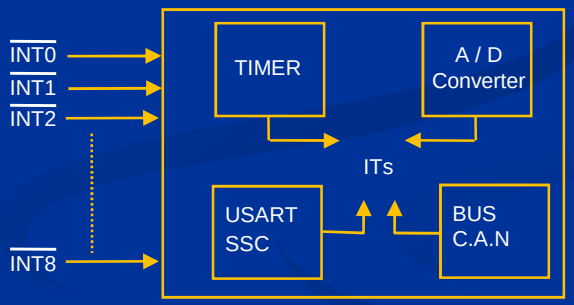
- **Masquable**

  Une interruption masquable peut être mise hors service par le microcontroleur

- **Non Masquable**

  Une interruption non masquable (NMI) ne peut être ignorée

- **Externe, Interne**



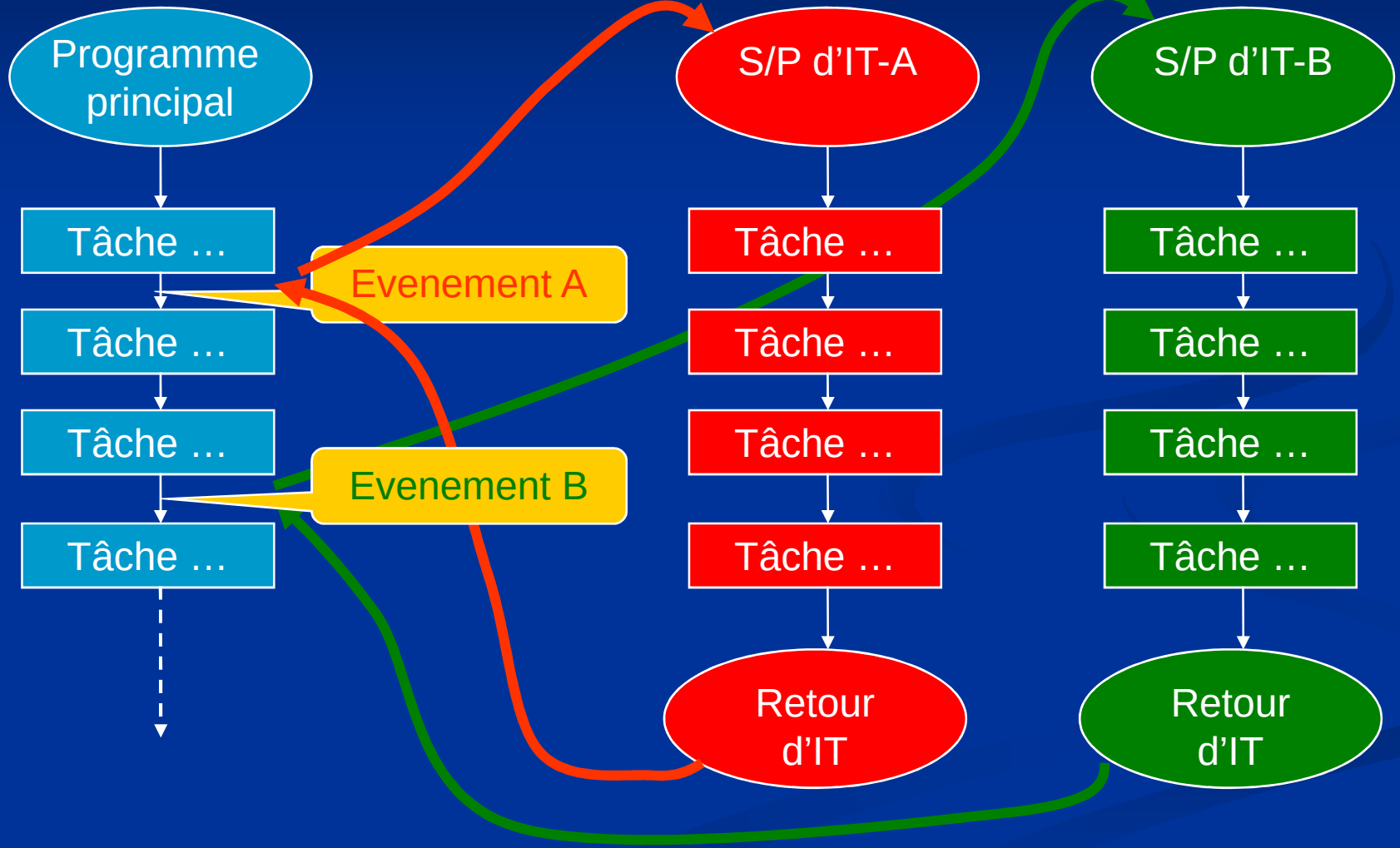
- **Acquittement**

  - **Matériel**
  - **Logiciel**

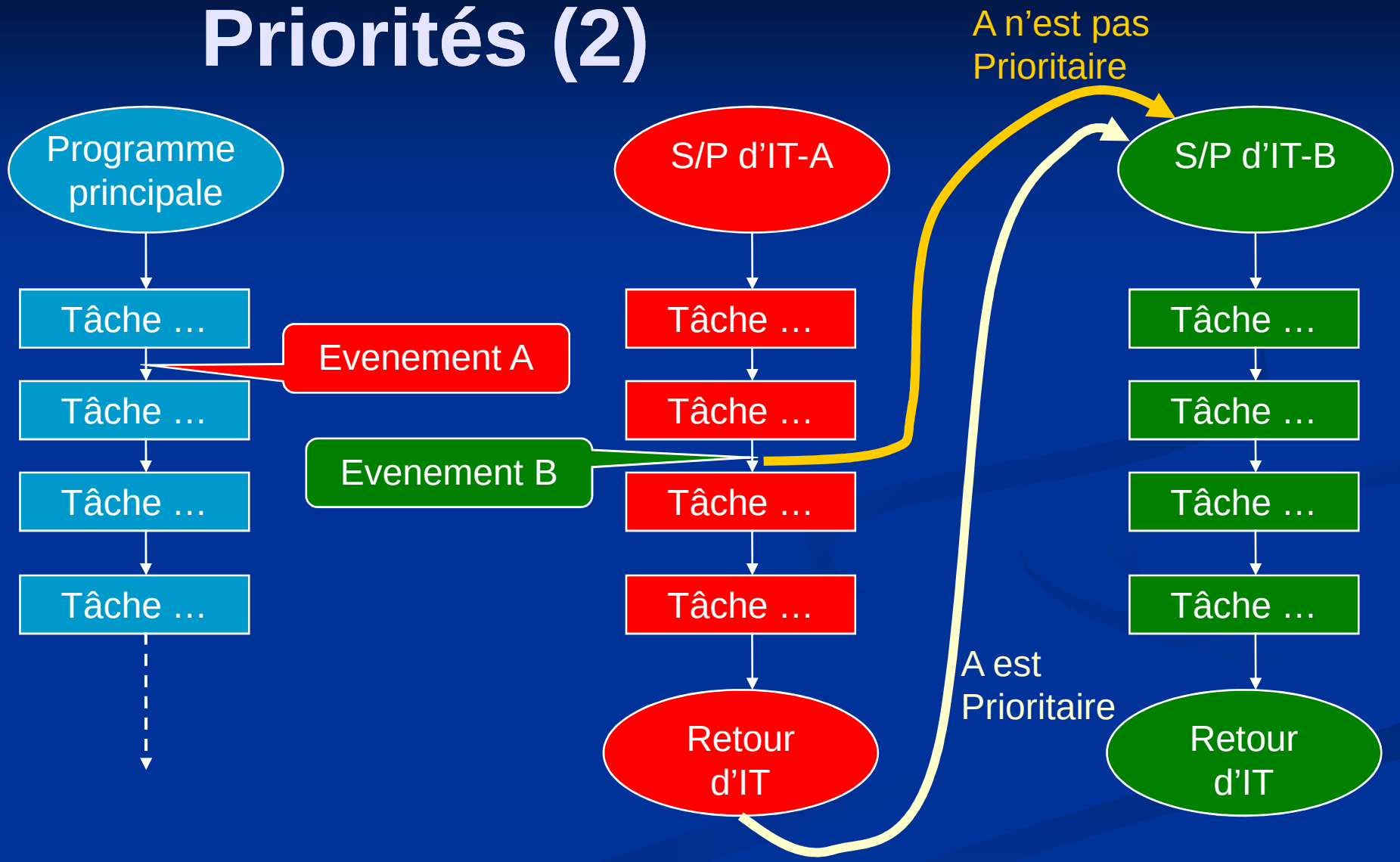
- **Vecteur**

  N° Vecteur ----> Vecteur

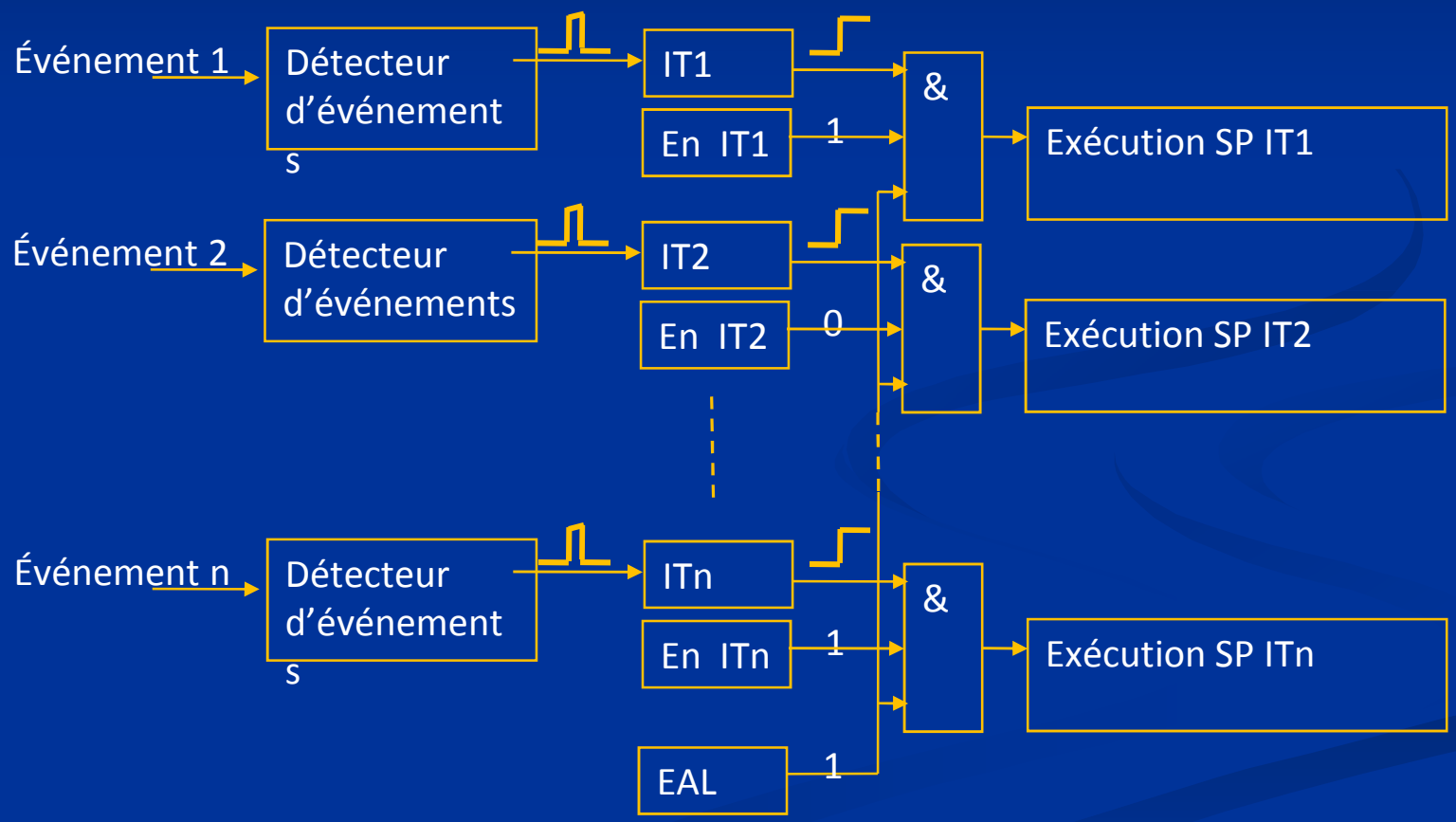
Interruptions – Priorités (1)



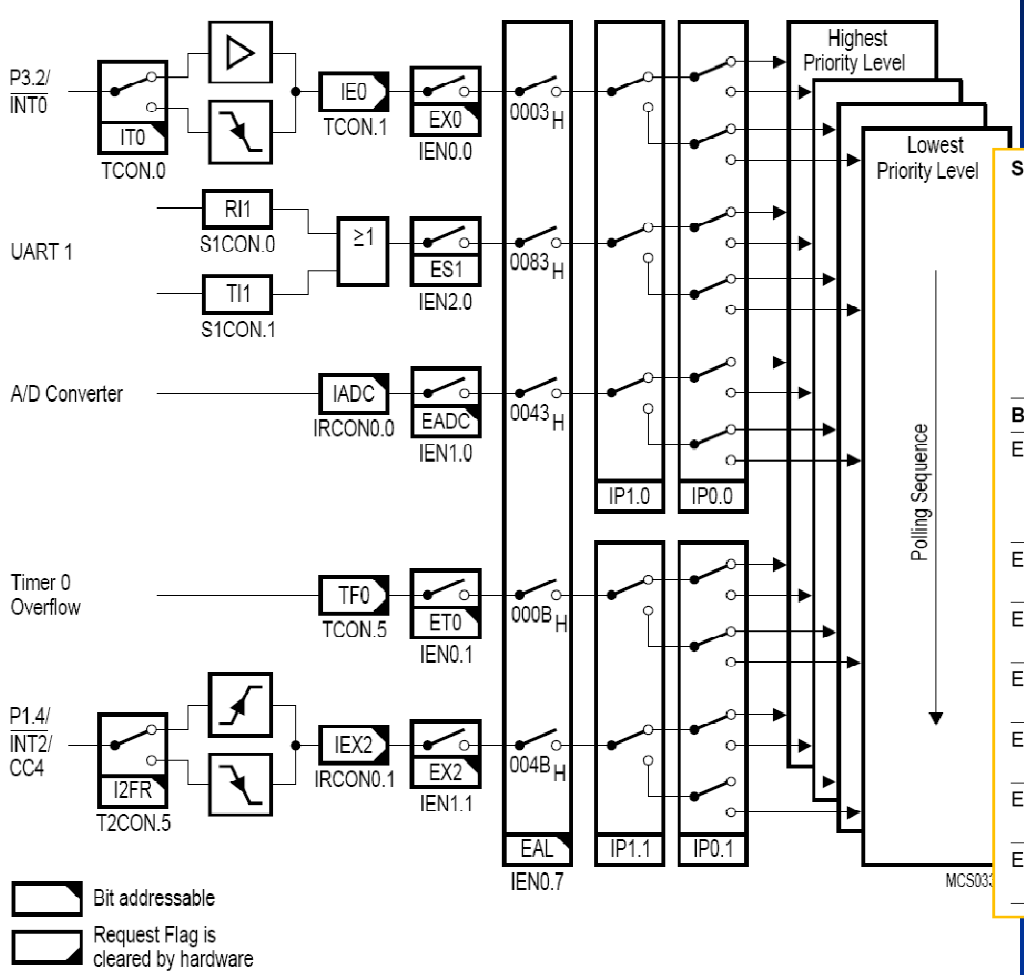
Interruptions – Priorités (2)



Masquage / Validation



IT sur C517A



Special Function Register IEN0 (Address $A8_H$) **Reset Value : 00_H**

Bit No.	MSB							LSB	
	AF_H	AE_H	AD_H	AC_H	AB_H	AA_H	$A9_H$	$A8_H$	
$A8_H$	EAL	WDT	ET2	ES0	ET1	EX1	ET0	EX0	IEN0

The shaded bit is not used for interrupt control

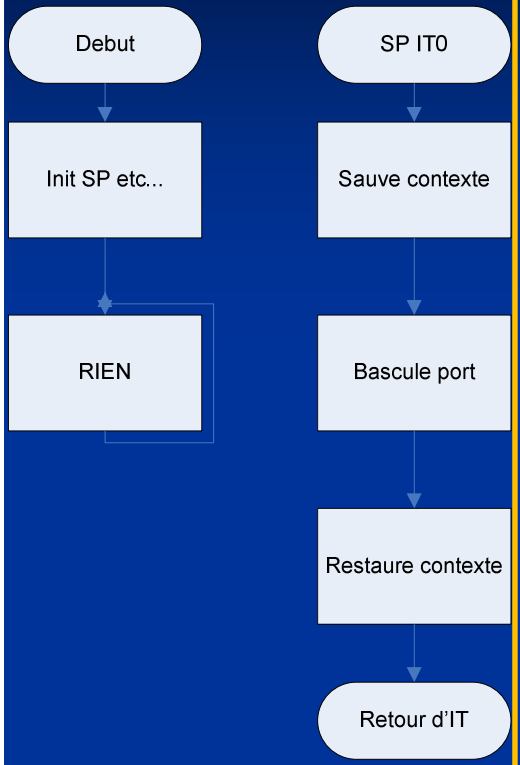
Bit	Function
EAL	Enable/disable all interrupts. If EA=0, no interrupt will be acknowledged. If EA=1, each interrupt source is individually enabled or disabled by setting or clearing its enable bit.
ET2	Timer 2 interrupt enable. If ET2 = 0, the timer 2 interrupt is disabled.
ES0	Serial channel 0 interrupt enable If ES0 = 0, the serial channel interrupt 0 is disabled.
ET1	Timer 1 overflow interrupt enable. If ET1 = 0, the timer 1 interrupt is disabled.
EX1	External interrupt 1 enable. If EX1 = 0, the external interrupt 1 is disabled.
ET0	Timer 0 overflow interrupt enable. If ET0 = 0, the timer 0 interrupt is disabled.
EX0	External interrupt 0 enable. If EX0 = 0, the external interrupt 0 is disabled.

Sources et vecteurs d'IT du C517A

8051 générique

Interrupt Source	Interrupt Vector Address	Interrupt Request Flags
External Interrupt 0	0003H	IE0
Timer 0 Overflow	000BH	TF0
External Interrupt 1	0013H	IE1
Timer 1 Overflow	001BH	TF1
Serial Channel 0	0023H	RI0 / TI0
Timer 2 Overflow / Ext. Reload	002BH	TF2 / EXF2
A/D Converter	0043H	IADC
External Interrupt 2	004BH	IEX2
External Interrupt 3	0053H	IEX3
External Interrupt 4	005BH	IEX4
External Interrupt 5	0063H	IEX5
External Interrupt 6	006BH	IEX6
Serial Channel 1	0083H	RI1 / TI1
Compare Match Interupt of Compare Registers CM0-CM7 assigned to Timer 2	0093H	ICMP0 - ICMP7
Compare Timer Overflow	009BH	CTF
Compare Match Interupt of Compare Register COMSET	00A3H	ICS
Compare Match Interupt of Compare Register COMCLR	00ABH	ICR

Programme d'IT : demoIT.a51

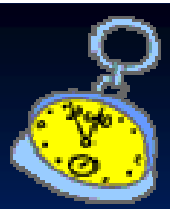



```
CSEG    AT      0
        ljmp    prog

        ORG     3       ; vecteur INT0
        push    acc     ; pour demo
        push    psw
        cpl     P2.0    ; bascule P2.0
        clr     IE0     ; efface drapeau
        pop     psw
        pop     acc
        reti

prog:   mov     sp,#7Fh
        setb    IT0     ; detection front
desc
        setb    EX0     ; active IT0
        setb    EAL   ;active les IT
        sjmp    $
        END
```

TP N°3

Essayer un programme fonctionnant en interruption
Créer un programme gérant plusieurs sources d'interruptions

TIMER / COMPTEUR



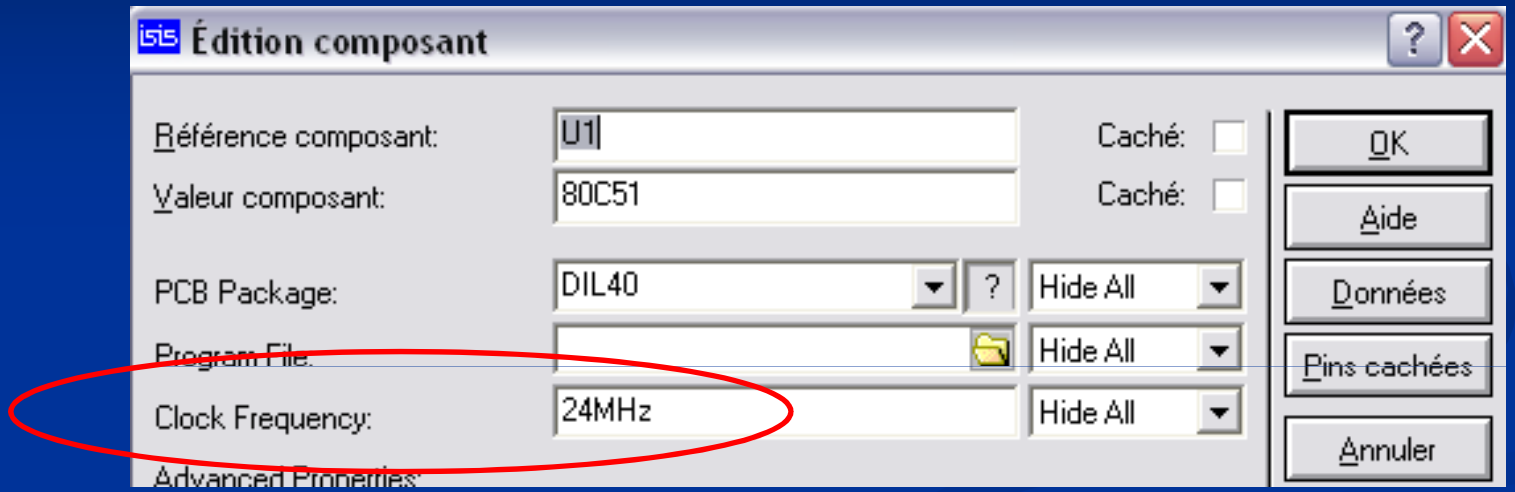
Les microcontrôleurs intègrent des compteurs (8 ou 16 bits) qui suivant leur utilisation sont nommé TIMER ou COMPTEUR.

Utilisation comme TIMER : généralement lorsque leur horloge est « constante » ils peuvent alors mesurer ou produire des « temps ».

Utilisation comme COMPTEUR : Généralement ils comptent des événements (fronts montants ou descendant) sur une entrée physique du microcontrôleur.

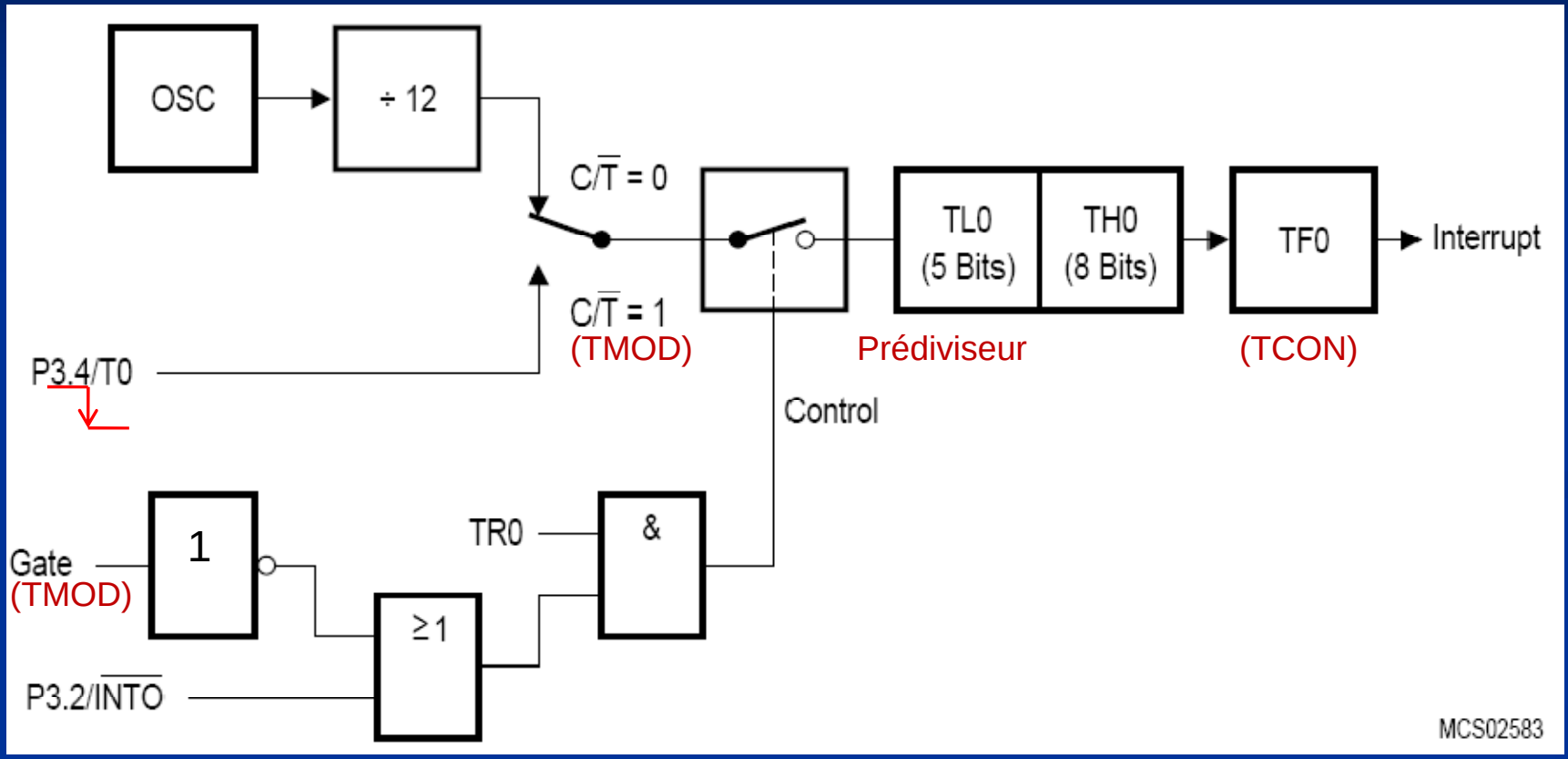
Le 8051 possède trois TIMER/COUNTER avec des applications différentes pour chaque

Choisir la fréquence de l'oscillateur

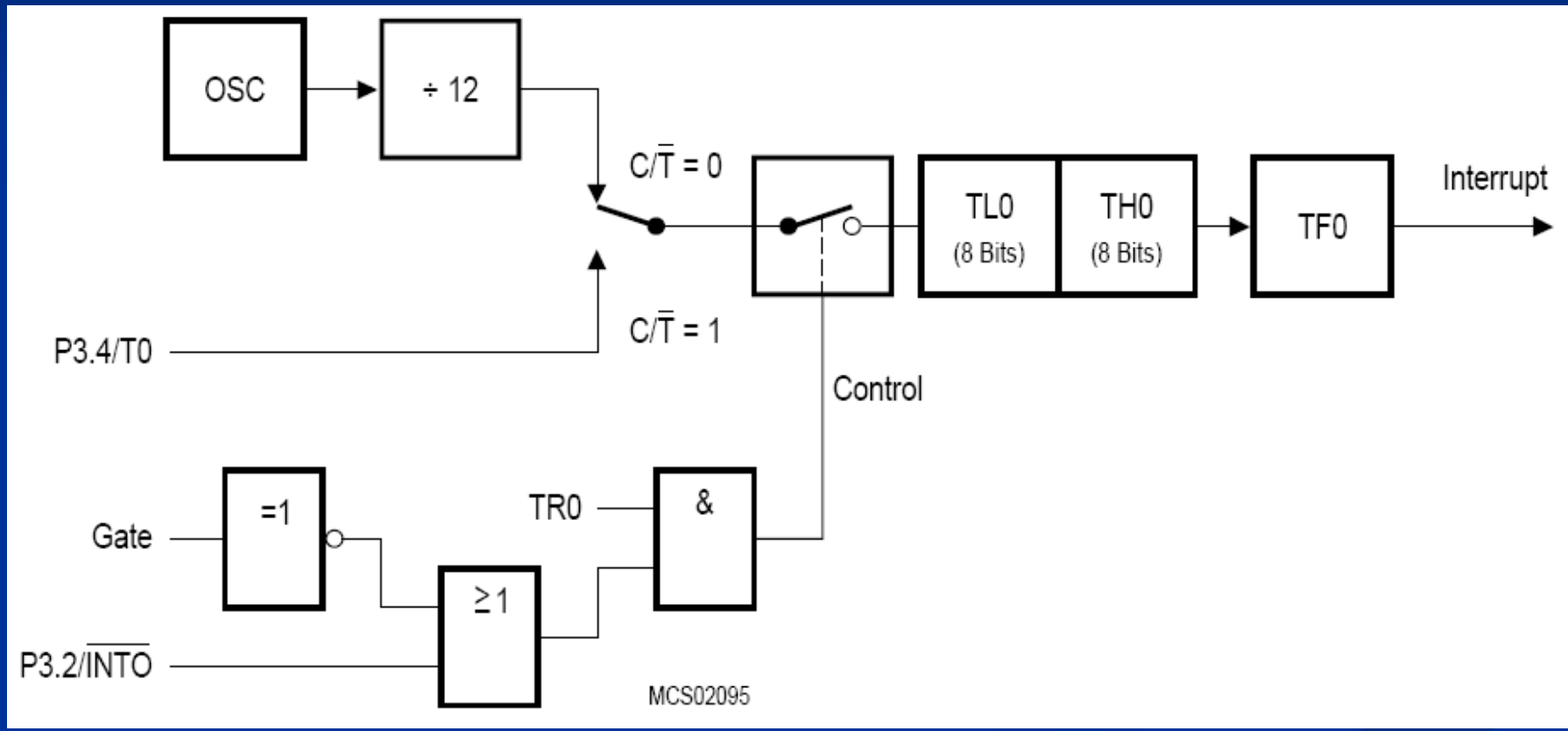


Lorsque ISIS-VSM est utilisé comme debugger, la fréquence de l'oscillateur n'est plus définie dans uVISION mais dans les propriétés ISIS du 8051 (clic-droit puis EDITER PROPRIETES)

MODE 0 pour TIMER 0 et 1



MODE1 pour TIMER 0 et 1



Le registre TCON

Special Function Register TCON (Address 88_H) **Reset Value : 00_H**

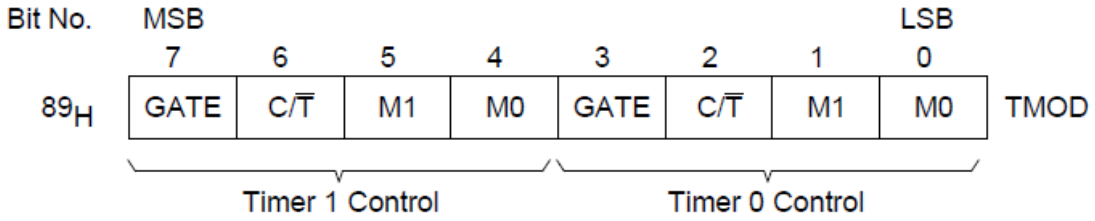
Bit No.	MSB 7	6	5	4	3	2	1	LSB 0	
	$8F_H$	$8E_H$	$8D_H$	$8C_H$	$8B_H$	$8A_H$	89_H	88_H	
88_H	TF1	TR1	TF0	TR0	IE1	IT1	IE0	IT0	TCON

The shaded bits are not used in controlling timer/counter 0 and 1.

Bit	Function
TR0	Timer 0 run control bit Set/cleared by software to turn timer/counter 0 ON/OFF.
TF0	Timer 0 overflow flag Set by hardware on timer/counter overflow. Cleared by hardware when processor vectors to interrupt routine.
TR1	Timer 1 run control bit Set/cleared by software to turn timer/counter 1 ON/OFF.
TF1	Timer 1 overflow flag Set by hardware on timer/counter overflow. Cleared by hardware when processor vectors to interrupt routine.

Le registre TMOD

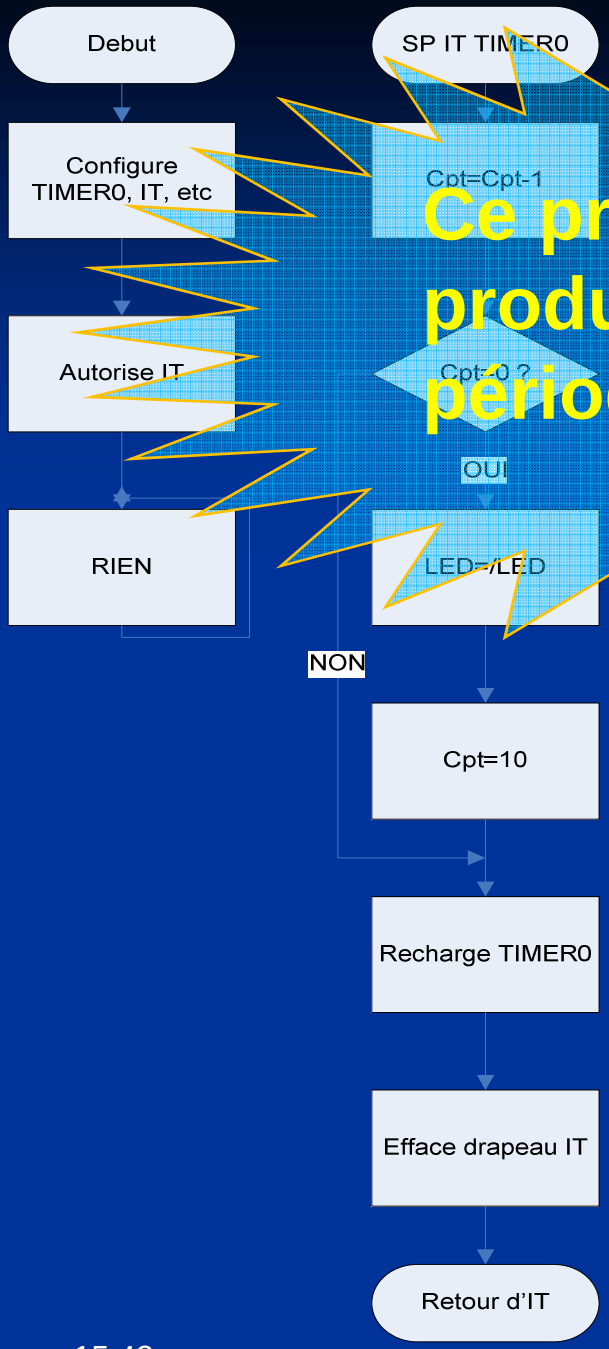
Special Function Register TMOD (Address 89_H) **Reset Value : 00_H**



Bit	Function
GATE	Gating control When set, timer/counter "x" is enabled only while "INT x" pin is high and "TRx" control bit is set. When cleared timer "x" is enabled whenever "TRx" control bit is set.
C/$\overline{T}$	Counter or timer select bit Set for counter operation (input from "Tx" input pin). Cleared for timer operation (input from internal system clock).
M1 M0	Mode select bits

M1	M0	Function
0	0	8-bit timer/counter: "THx" operates as 8-bit timer/counter "TLx" serves as 5-bit prescaler
0	1	16-bit timer/counter. "THx" and "TLx" are cascaded; there is no prescaler
1	0	8-bit auto-reload timer/counter. "THx" holds a value which is to be reloaded into "TLx" each time it overflows
1	1	Timer 0 : TL0 is an 8-bit timer/counter controlled by the standard timer 0 control bits. TH0 is an 8-bit timer only controlled by timer 1 control bits. Timer 1 : Timer/counter 1 stops

Exemple mode 1: flash_it.a51

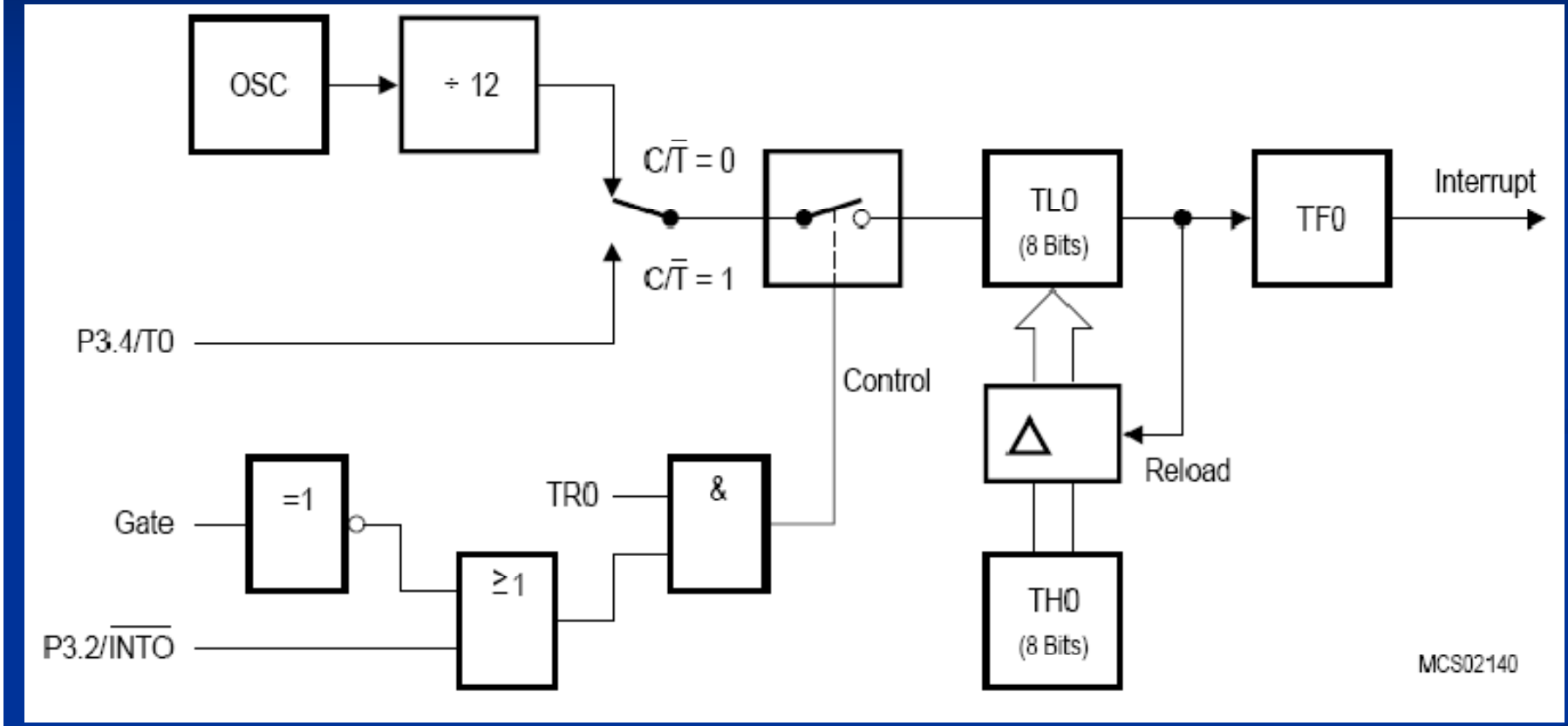


Ce programme ne produira pas une période de 500ms !

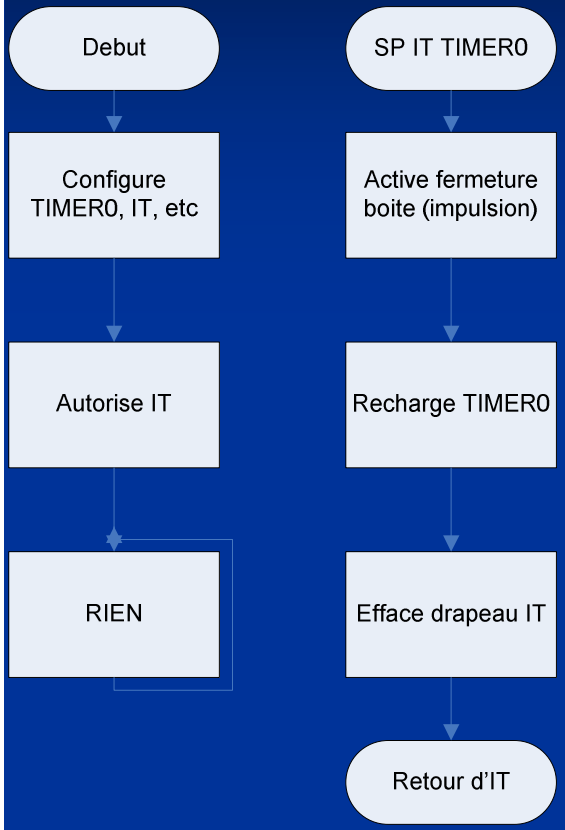
```
valTH0  EQU 3Ch ; 3CB0 = FFFFh-50000d+1
valTL0  EQU 0B0h
LED     EQU P2.0
        CSEG    AT 0
        LJMP    PROG
        ORG     0x000B  ; vecteur TIMER0 ovf
        LJMP    IT_TIMER0
PROG:   MOV     SP,#STACK-1
        MOV     TMOD,#00000001b
        SETB    TR0     ; run TIMER 0
        MOV     CPT,#10     ; compteur d'IT
        SETB    ET0     ; IT sur OVF TIMER0
        SETB    EAL     ;
        SJMP    $       ; attend ;
IT_TIMER0:
        DJNZ    CPT,NON
        CPL     LED     ; bascule LED
        MOV     CPT,#10 ;recharge cpt d'IT
NON:    MOV     TH0,#3Ch    ; recharge TIMER0
        MOV     TL0,#0AFh
        CLR     TF0     ; efface drapeau IT0
        RETI
```

FLASH sur P2.0, BdT TIMER0 mode 1 , periode 0,5s (demi periode 250 mS)
Bdt=500nS : comptage : 50000 pulses soit 25mS,
FFFFh-50000d= 15535d = 3CAFh
Bascule toutes les 10 interruptions soit 250mS

MODE 2 pour TIMER 0 et 1



Exemple 2 Comptage d'événements demo_cpt_MODE1.A51

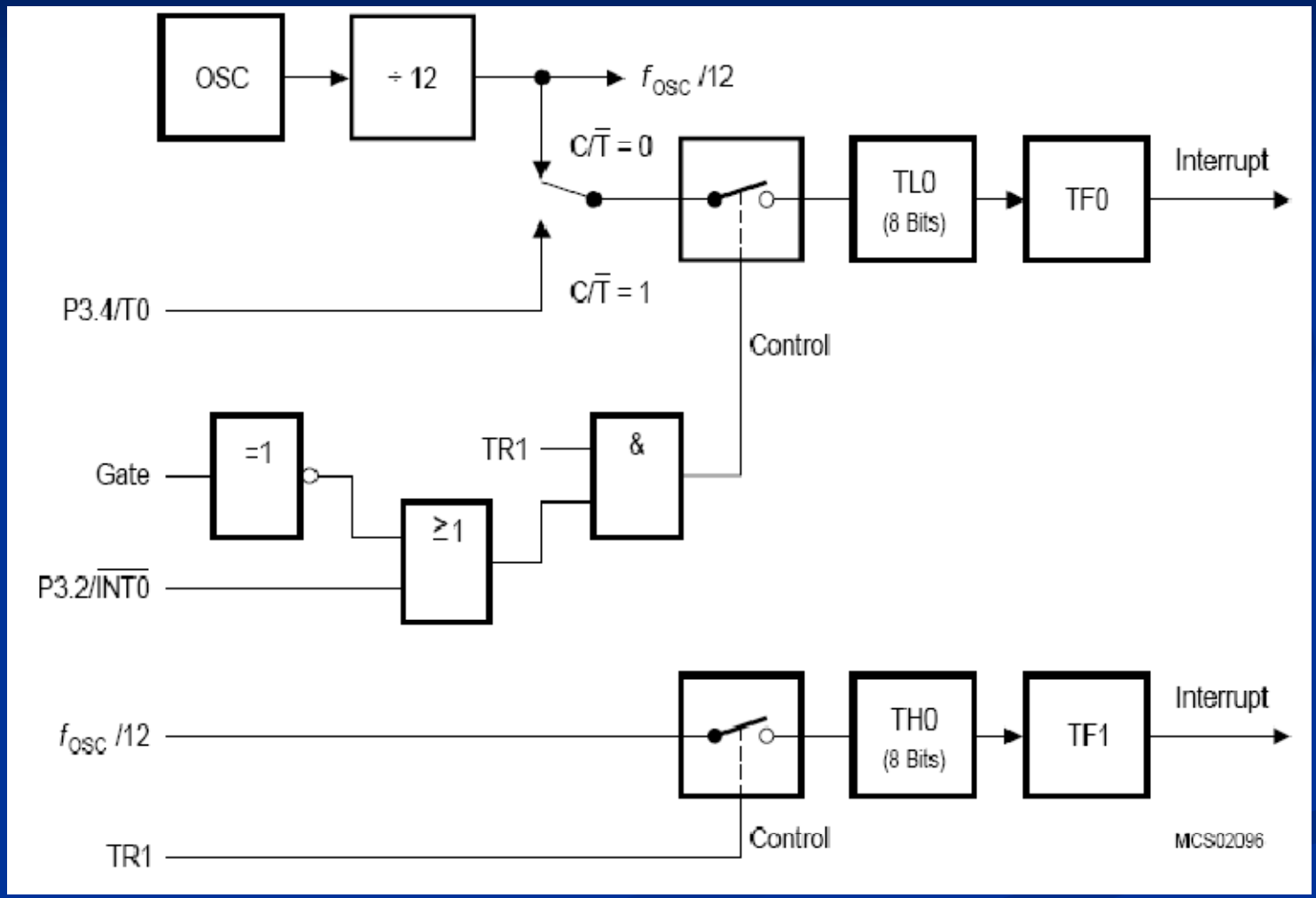



```
; 10 bonbons par boite
valTH0 EQU       0FFh
valTL0 EQU       0F6h
BOITE  EQU       P2.0 ; fermeture boite
STACK  SEGMENT   IDATA
       RSEG      STACK
       DS        32
CPT:   DS        1       ; compteur
       CSEG    AT 0
       LJMP    start
       ORG 0x000B     ; vecteur T0 ovf
       LJMP   IT_TIMER0
…
```

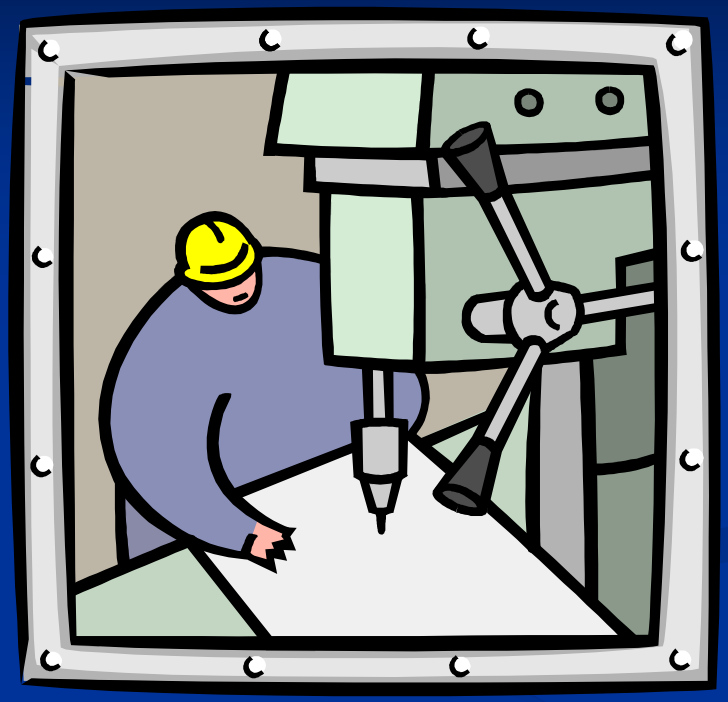
Un capteur optique détecte le passage de bonbons sur un tapis roulant les entrainant dans une boite. La boite doit être fermée, lorsqu'elle contient dix bonbons.

```
start:          MOV    SP,#STACK-1
                MOV    TMOD,#00000101b ; TIMER 0 sur mode 1
source Q/12
                MOV    CPT,#0 ; compteur de boites =0
                MOV    TH0,#valTH0  ; recharge TIMER0
                MOV    TL0,#valTL0
                SETB   ET0    ; autorise IT sur TIMER0
                SETB   EAL    ; autorise toutes les IT
                SETB   TR0    ; run TIMER 0
                SJMP   $      ; attend IT
;
IT_TIMER0:      SETB   BOITE  ; impulsion sur fermeture
                CLR    BOITE
                INC    CPT    ;incrémente compteur de boites
                MOV    TH0,#valTH0  ; recharge TIMER0
                MOV    TL0,#valTL0
                CLR    TF0    ; efface drapeau IT0
                RETI
               END              ; End Of File
```

MODE 3

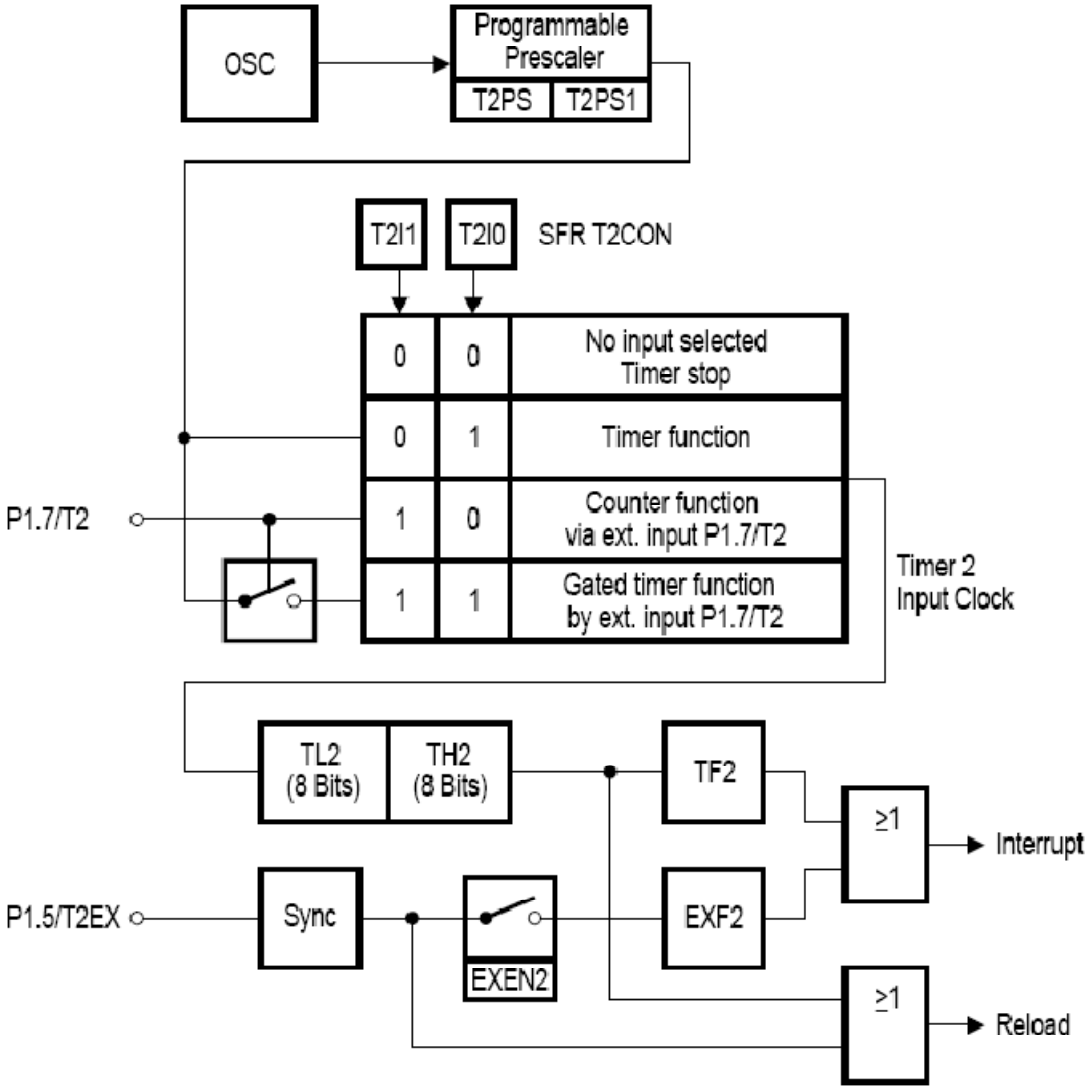

OSC
÷ 12
f_{osc} /12
C/T = 0
C/T = 1
P3.4/T0
TL0
(8 Bits)
TF0
Interrupt
Control
Gate
=1
TR1
&
≥1
P3.2/INT0
f_{osc} /12
TH0
(8 Bits)
TF1
Interrupt
TR1
Control
MCS02096

TP N°4



Mise en œuvre des TIMERS
comptage
production de signaux

TIMER 2



Les fonctions matérielles CAPTURE/COMPARE permettent de limiter considérablement le logiciel. **COMPARE** automatisent les actions sur les broches externes. (disponible sur certains 8051)
CAPTURE facilite l'échantillonnage temporel pour la mesure de durée.

Les fonctions CAPTURE/COMPARE s'appuient sur le TIMER 2 ou le « COMPARE TIMER »

T2I1, T2I0 =0,1 le TIMER 2 compte l'horloge issue du prédiviseur OSC/12 /24 /48 /96
T2I1, T2I0 =1,0 le COUNTER 2 compte les fronts descendants sur T2
T2I1, 0T2I0 =1,1 le TIMER 2 compte l'horloge issue du prédiviseur OSC/12 (uniquement) lorsque T2=1
En cas de débordement TF2=1 une interruption peut être activée.

TIMER2
Le registre TCON2

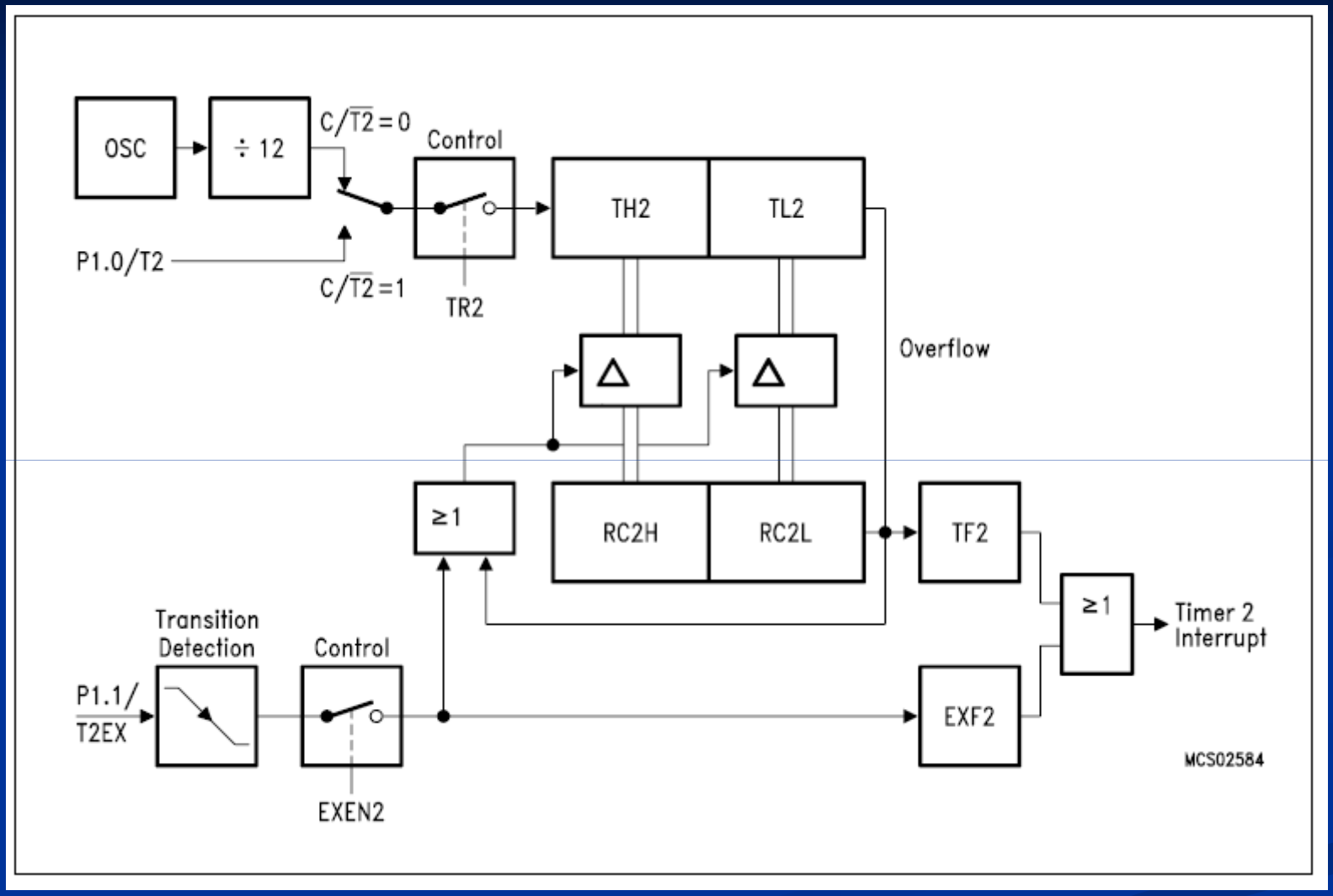
Special Function Register T2CON (Address $C8_H$) **Reset Value : 00_H**

Bit No.	MSB 7	6	5	4	3	2	1	LSB 0	
	CF_H	CE_H	CD_H	CC_H	CB_H	CA_H	$C9_H$	$C8_H$	
$C8_H$	TF2	EXF2	RCLK	TCLK	EXEN2	TR2	C/$\overline{T2}$	CP/$\overline{RL2}$	T2CON

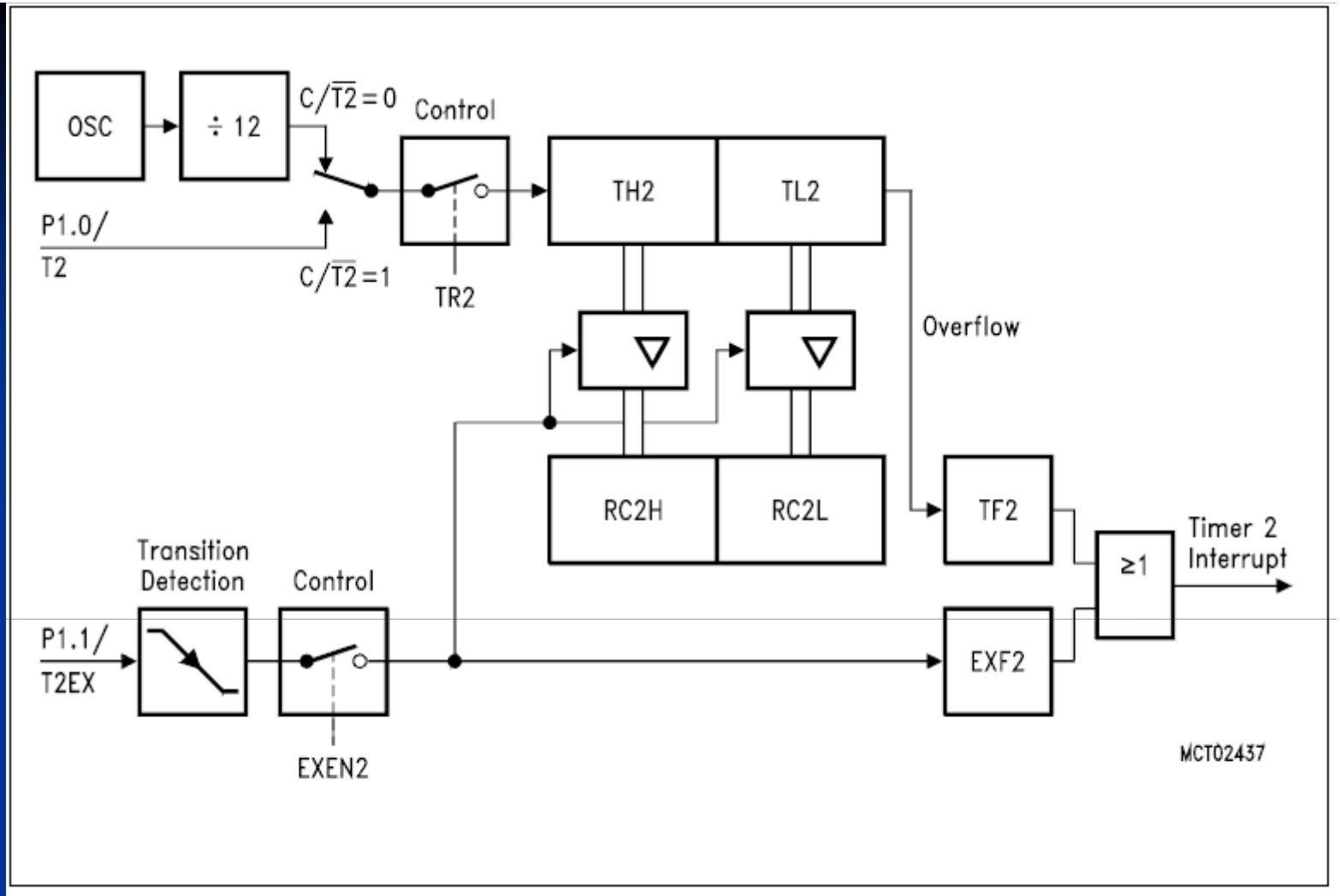
Bit	Function
TF2	Timer 2 Overflow Flag Set by a timer 2 overflow. Must be cleared by software. TF2 will not be set when either RCLK = 1 or TCLK = 1.
EXF2	Timer 2 External Flag Set when either a capture or reload is caused by a negative transition on T2EX and EXEN2 = 1. When timer 2 interrupt is enabled, EXF2 = 1 will cause the CPU to vector to the timer 2 interrupt routine. EXF2 must be cleared by software. EXF2 does not cause an interrupt in up/down counter mode (DCEN = 1, SFR T2MOD)
RCLK	Receive Clock Enable When set, causes the serial port to use timer 2 overflow pulses for its receive clock in serial port modes 1 and 3. RCLK = 0 causes timer 1 overflows to be used for the receive clock.
TCLK	Transmit Clock Enable When set, causes the serial port to use timer 2 overflow pulses for its transmit clock in serial port modes 1 and 3. TCLK = 0 causes timer 1 overflow to be used for the transmit clock.
EXEN2	Timer 2 External Enable When set, allows a capture or reload to occur as a result of a negative transition on pin T2EX (P1.1) if timer 2 is not being used to clock the serial port. EXEN2 = 0 causes timer 2 to ignore events at T2EX.
TR2	Start / Stop Control for Timer 2 TR2 = 1 starts timer 2.
C/$\overline{T2}$	Timer or Counter Select for Timer 2 C/$\overline{T2}$ = 0 for timer function. C/$\overline{T2}$ = 1 for external event counter (falling edge triggered).
CP/$\overline{RL2}$	Capture /Reload Select CP/$\overline{RL2}$ = 1 causes captures to occur an negative transitions at pin T2EX if EXEN2 = 1. CP/$\overline{RL2}$ = 0 causes automatic reloads to occur when timer 2 overflows or negative transitions occur at pin T2EX when EXEN2 = 1. When either RCLK = 1 or TCLK = 1, this bit is ignored and the timer is forced to auto-reload on timer 2 overflow.

RCLK + TCLK	CP/$\overline{RL2}$	TR2	Mode
0	0	1	16-bit Auto-Reload
0	1	1	16-bit Capture
1	X	1	Baud Rate Generator
X	X	0	(off)

TIMER mode AUTO-RELOAD

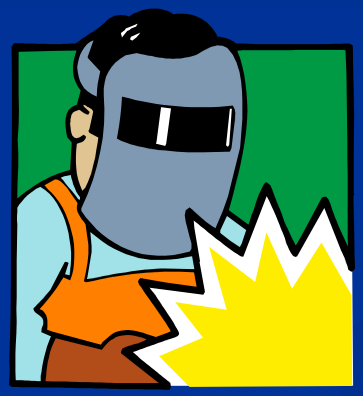


CAPTURE

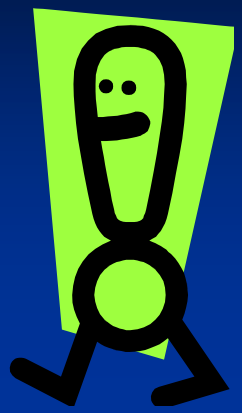


Lors de l'événement (front montant ou descendant) sur une broche le contenu du TIMER 2 est recopié dans un registre, une interruption peut être générée. Le microcontrôleur conserve ainsi une trace très précise de l'instant ou s'est produit l'événement. Le traitement et les actions correspondantes peuvent être traitées ensuite.

TP N°5



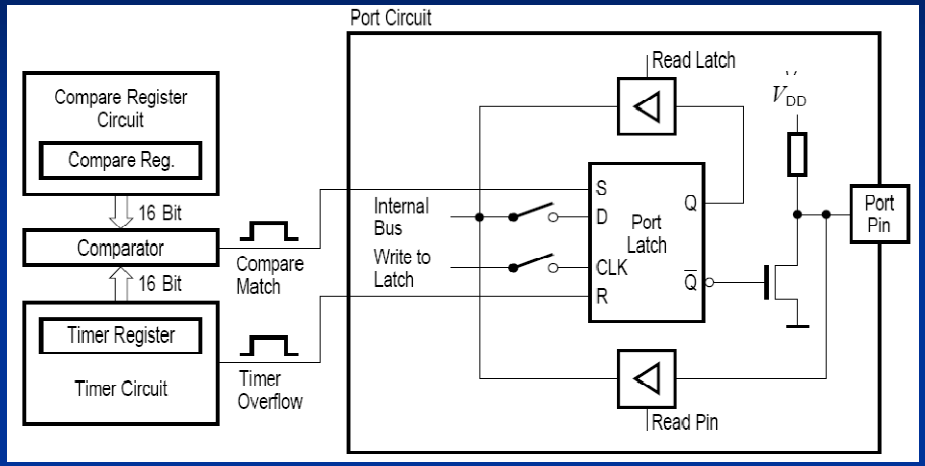
CAPTURE,
Mesures de durée
Mesures de périodes

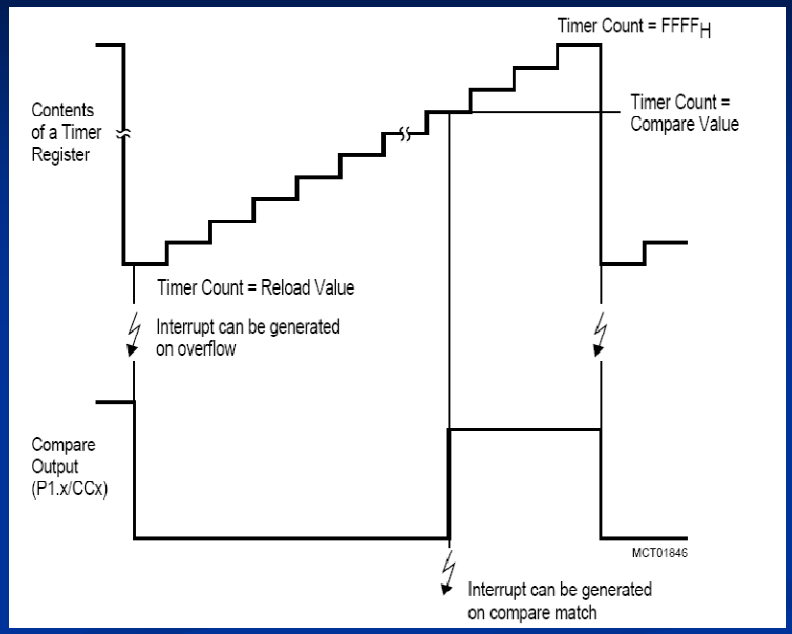


Les fonctionnalités TIMER/COUNTER suivantes **n'existent par sur tous les modèles 8051**

N'existe pas sur les 8051 génériques

C517A - COMPARE AVEC TIMER 2 : MODE 0



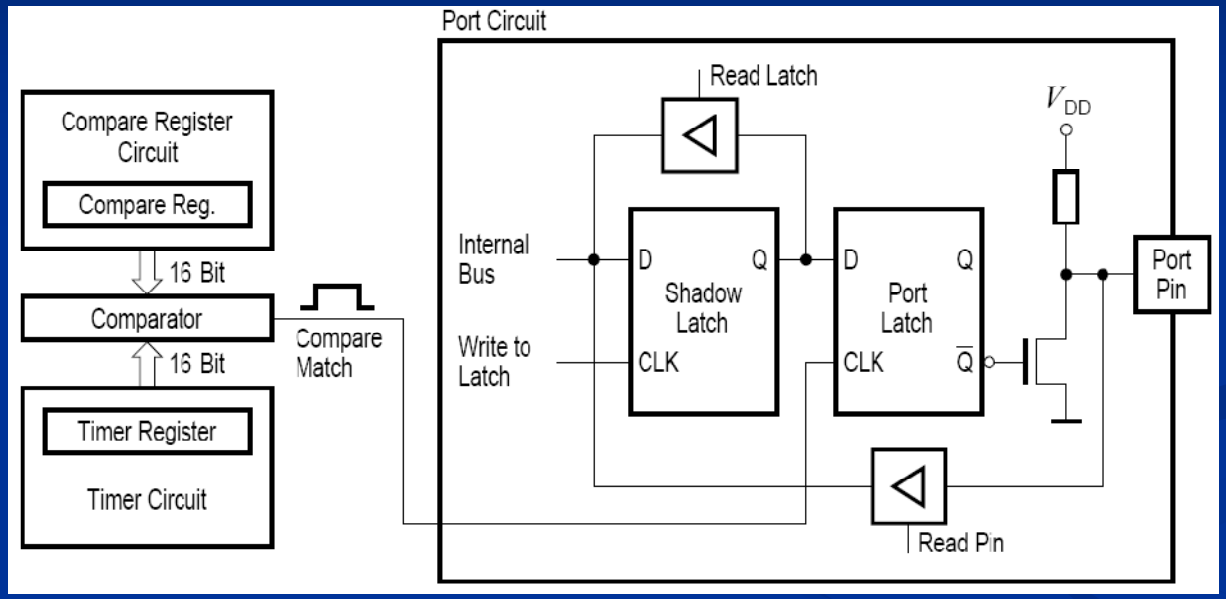


N'existe pas sur les 8051 génériques

Lorsqu'il y a coïncidence (match) entre le TIMER 2 et le registre de comparaison (CCx) la broche sélectionnée est mise à 1. (S=1 ⇨ Q=1 et /Q=0 ⇨ le transistor est bloqué ⇨ le pull-up ramène un NL1 sur la sortie)
Lorsque le TIMER déborde la broche sélectionnée est mise à 0 et le TIMER peut être rechargé.
Ce principe est particulièrement adapté à la modulation de largeur d'impulsion, Pulse Wave Modulation (PWM) pour le contrôle de vitesse des moteurs ou la synthèse vocale .

C517A - COMPARE AVEC TIMER 2 : MODE 1

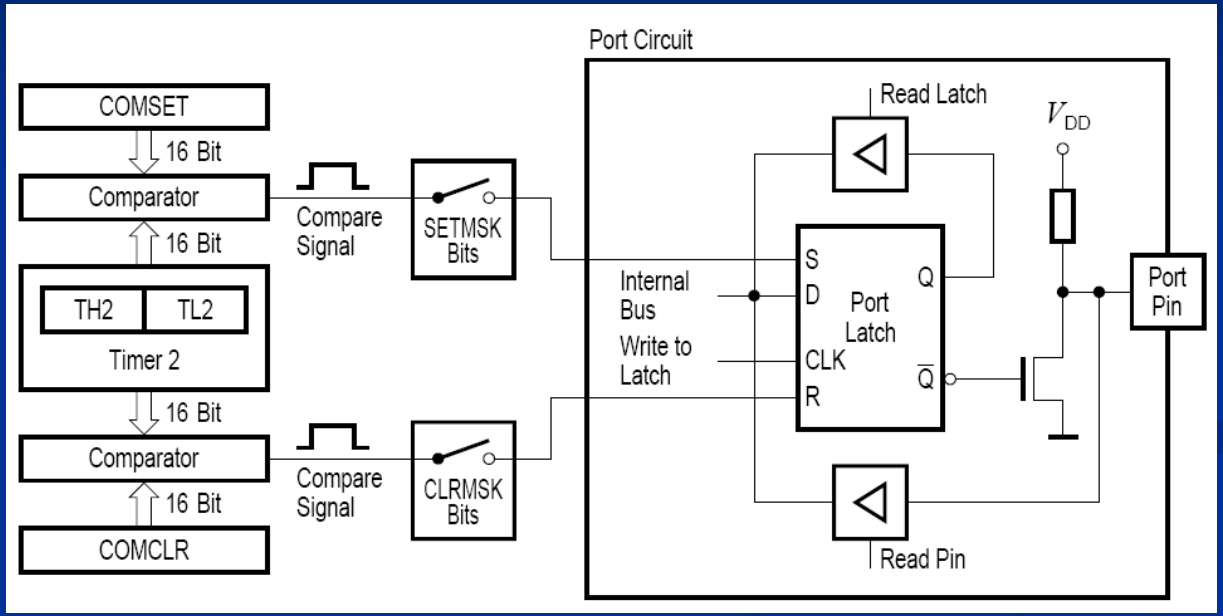
N’existe pas sur les 8051 génériques



A chaque coïncidence la donnée dans le Latch de sortie est recopiée sur le port. Cette donnée est choisie par programmation et placée sur le Latch interne. Il donc est possible de choisir le niveau de sortie lors de la coïncidence

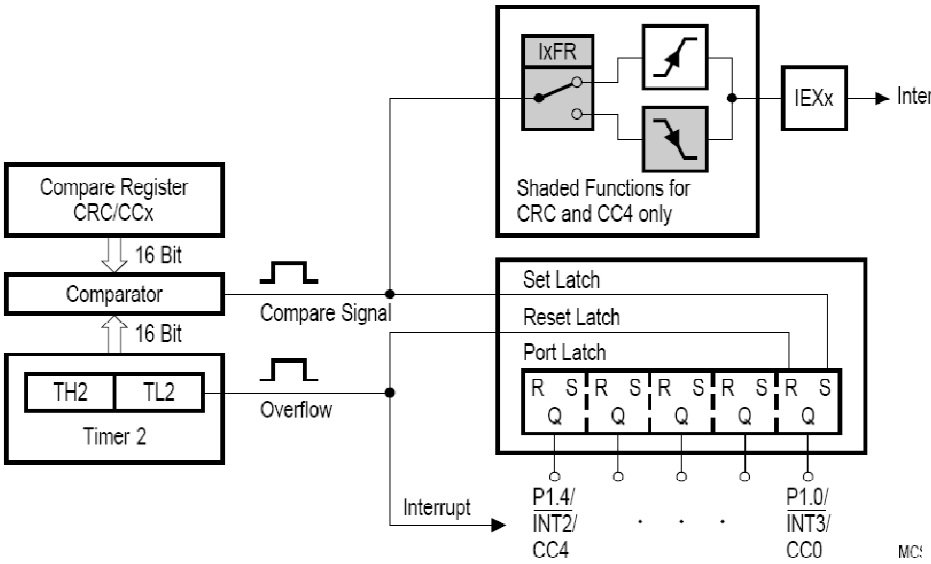
C517A - COMPARE AVEC TIMER 2 : MODE 2

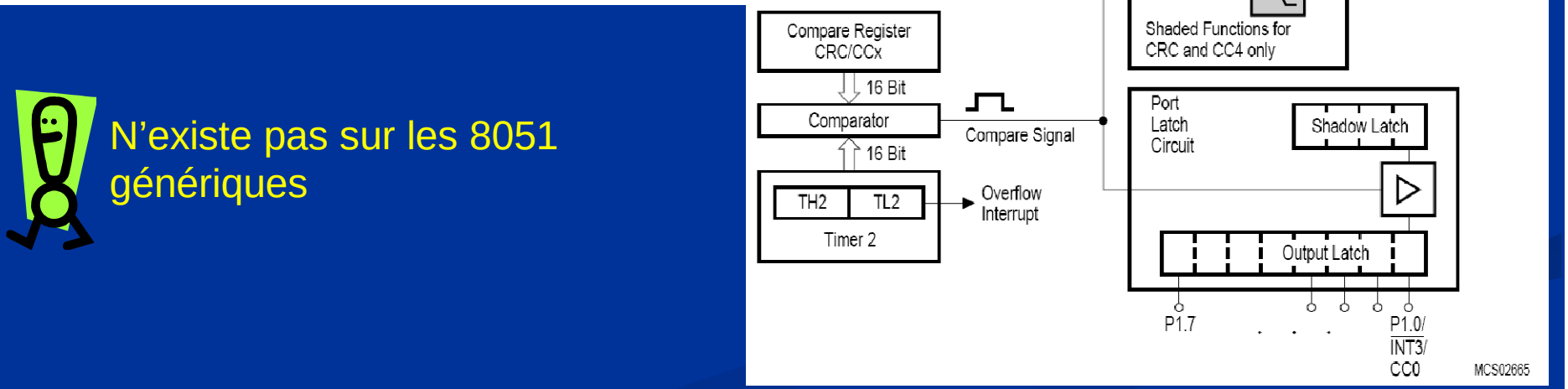
N'existe pas sur les 8051 génériques



Quand il y a coïncidence entre le TIMER 2 et le registre COMSET un NL1 apparaît sur le port en sortie si le bit de masque SETMSK est à 1
Quand il y a coïncidence entre le TIMER 2 et le registre COMCLR un NL0 apparaît sur le port en sortie si le bit de masque CLRMSK est à 1

C517A - Commande simultanée de plusieurs broches en MODE 0 et 1

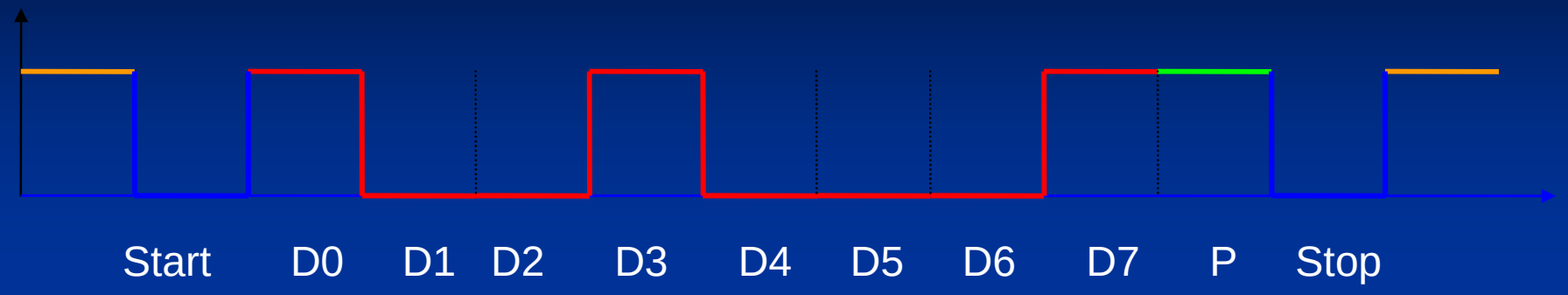


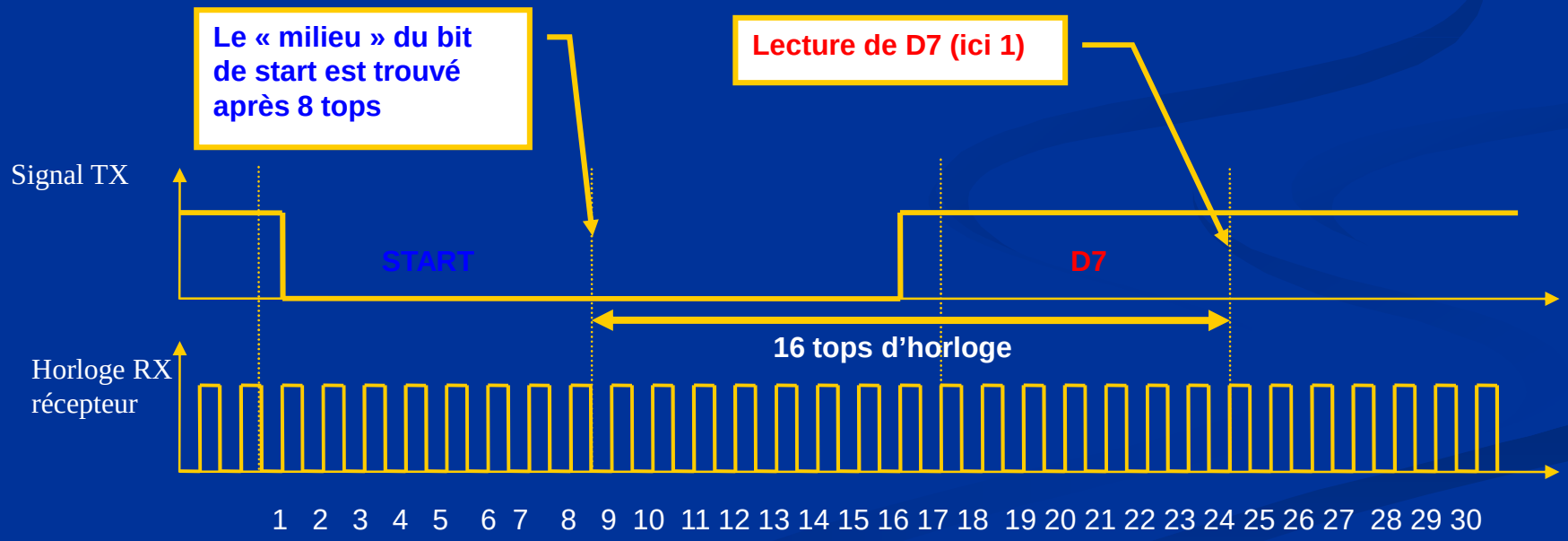


N’existe pas sur les 8051 génériques

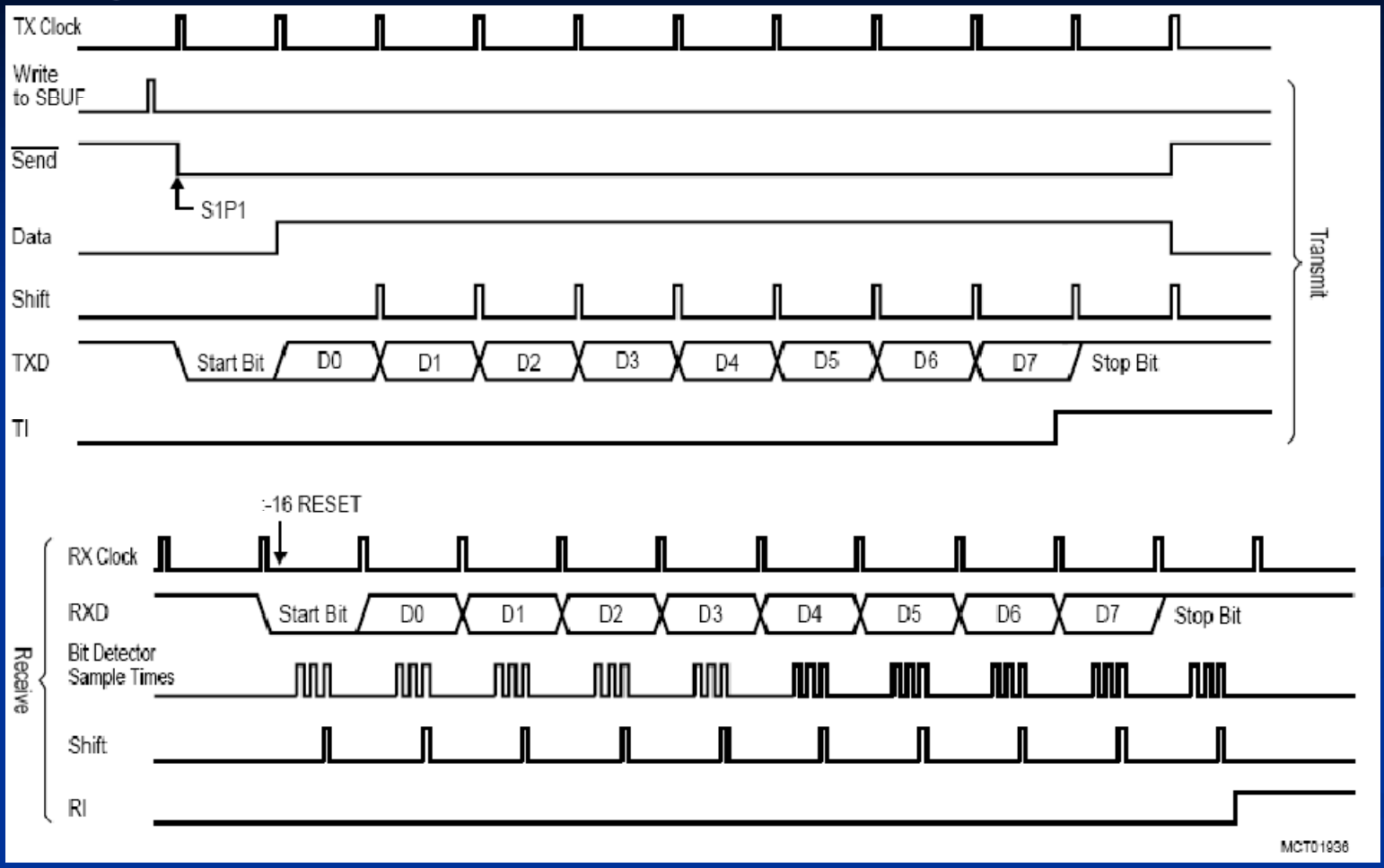
Communications asynchrones

PORTS SERIES ASYNCHRONES



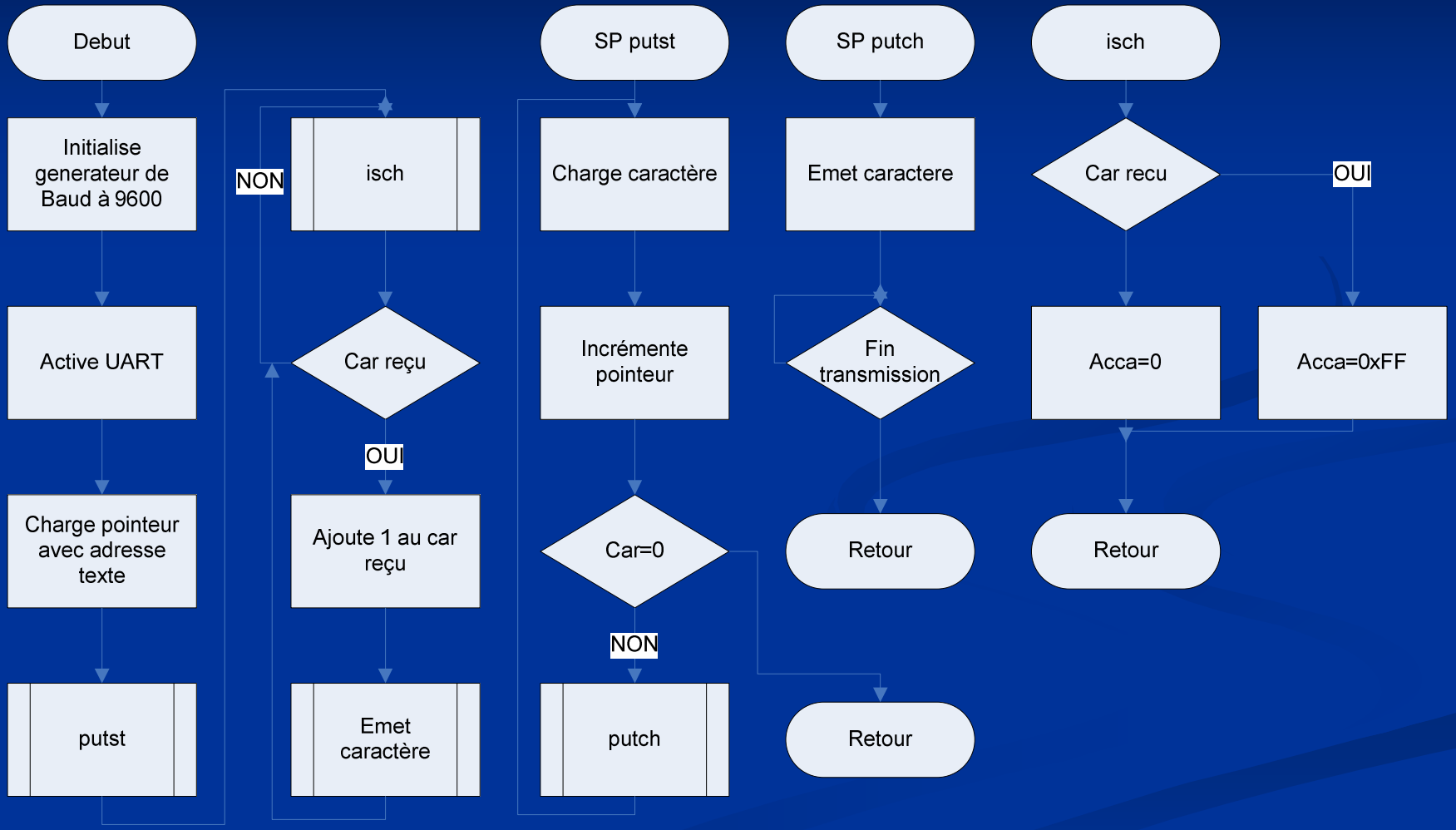


UART



Send est un signal interne à l'USART qui valide la sortie des données
Shit est un signal interne commandant le décalage du registre de transmission ou réception
Une transmission est initiée par une écriture dans SxBUF. A la fin le bit TI passe à 1 et peut générer une interruption, ce bit doit être remis à zéro par logiciel.
Une réception est initiée par la réception d'un bit de START. A la fin de la réception le bit RI passe à 1 et peut générer une interruption, ce bit doit être remis à zéro par logiciel. L'octet reçu se trouve dans SxBUF.

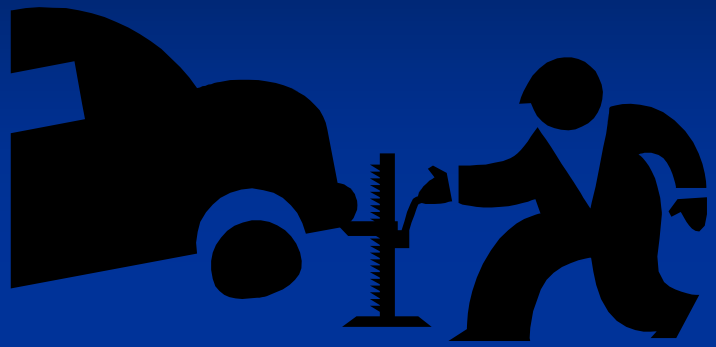
Communications asynchrones : exemple



```
#include <reg517.inc>
sorelh      equ 0BAh; registres baud
sorell      equ 0AAh
TIO     equ S0CON.1  ; bit fin reception
RIO     equ S0CON.0  ; bit fin reception
stack       segment  IDATA    ; pour pile S
prog    segment  code; zone programme
pconst      segment  code; pour constantes
    RSEGstack
    DS  10h ; reserve 16 octets pour la pile
;
    CSEGAT 0; RAZ
    ljmpstart
;
    RSEGprog
start:      mov sp,#stack-1
    setbadcon0.7     ; FOSC = baud generator
    mov sorelh,#3        ;9600 Bauds (Q=24MHz)
    mov sorell,#0D9h
    mov S0CON,#01010000B ; mode 1, rx actif
    ljmptache
; SP emission d'un message (fin par 0)
putst:      mov R0,#0FFh; R0 est le pointeur de caractère
suite:      inc R0       ; caractère suivant
    mov a,R0     ; dans acc
    movca,@a+dptr
    jz  fin      ; si acc=0 c'est finit
    callputch        ; sinon émission caractère
    sjmpsuite        ; au suivant !
fin:    ret
```

```
; SP emission d'un caractère (dans acc)
putch:          mov     S0BUF,a  ; caractère dans registre de transmission
        jnb     TIO,$            ; transmission terminée ?
        clr     TIO              ; efface drapeau
        ret
; SP test caractere reçu
isch:           jnb     RIO,non  ; attend caractère
        mov     a,#0FFh          ; si oui acc=FF
        ret
non:            clr     a                ; si non acc=0
        reta
; SP reception caractere
getch:          clr     RIO              ; efface drapeau
        mov     a,S0BUF
        ret
; pour test, emission du message puis echo+1
tache:          mov     dptr,#txt        ; adresse message dans pointeur ROM
        call    putst            ; emission message
repete:         call    isch             ; attend reception
        jz      repete
        call    getch            ; récupère le caractère reçu
        inc     a                ; retourne car +1
        call    putch
        sjmp    repete           ; boucle sans fin
; message à transmettre
        RSEG    PCONST
txt:            db      'Test communications - ECHO+1',10,13,0
        END
```

TP N°6



Gestion des communications asynchrones
Emission, réception de données

FIN (provisoire)