

Assembleur ARM7TDMI 2

ZZ1

Christophe de Vaulx & Alain Tanguy

Architecture de l'ARM7TDMI

Historique

- 1983 La compagnie britannique Acorn Computers Ltd développe des processeurs ARM.
- 1985 Le premier processeur de cette famille l'ARM1 (Acorn Risc Machine 1) est disponible.
- Fin des années 80, Acorn s'associe avec Apple et VLSI pour mettre au point l'ARM6.
- 1990 Ils fondent Advanced Risc Machines Ltd.
- 1991 Sortie de L'ARM6 (Advanced Risc Machine 6).
- 1998 La société devient ARM Ltd

Généralités

- Aujourd'hui, les cœurs de processeurs ARM (ARM7, ARM9, ARM11...) représentent environ 75% du marché des processeurs 32 bits embarqués.
- Le terme "cœur" signifie que ces processeurs sont vendus comme des blocs à utiliser dans un circuit qui intègre d'autre blocs pour constituer un système sur puce "SoC".
- Leur succès s'explique par leur bon rapport encombrement / consommation / performances.

Quelques exemples de produits qui utilisent des cœurs ARM



- Nintendo DS (1 ARM7 et 1 ARM9)



- PSP de SONY (1 ARM9)

Quelques exemples de produits qui utilisent des cœurs ARM



- Kodak EasyShare LS743 Digital Zoom Camera (ARM7)



- Motorola RAZR V8 (ARM11)

Quelques exemples de produits qui utilisent des cœurs ARM

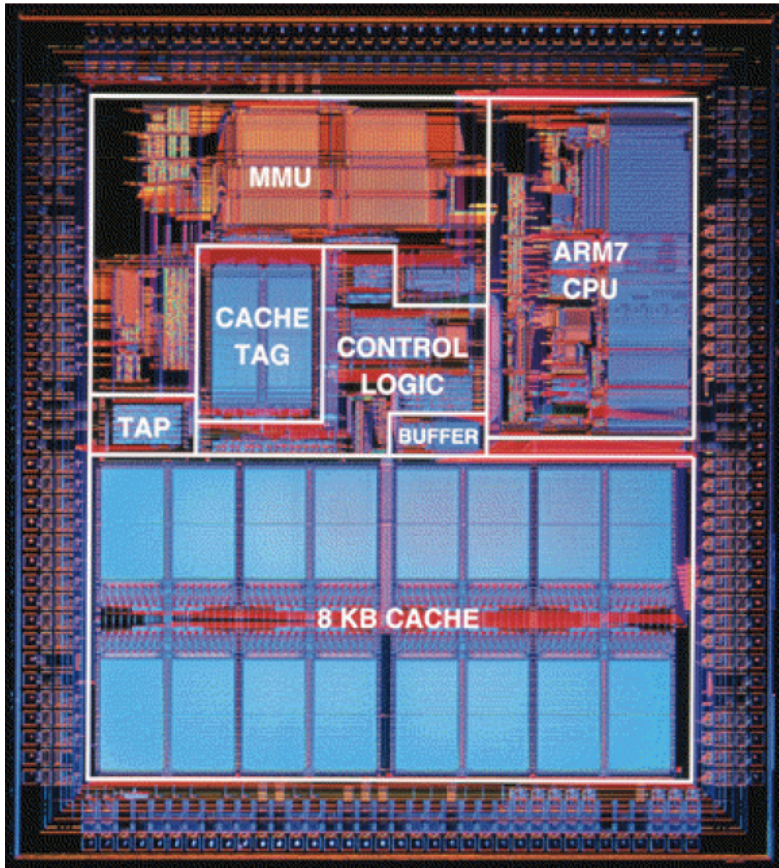


Principales caractéristiques de l'ARM7TDMI

- Processeur à Architecture « Von Neumann ».
- Pipeline 3 étages : Fetch, Decode et Execute.
- Deux jeux d'instructions : 32 bits (mode ARM) et 16 bits (mode Thumb).
- Instructions d'accès mémoire LOAD et STORE.
- T : support du mode Thumb.
- D : extensions pour la mise au point.
- M : Multiplieur 32x8 et instructions résultat 64 bits.
- I : émulateur embarqué ("Embedded ICE").

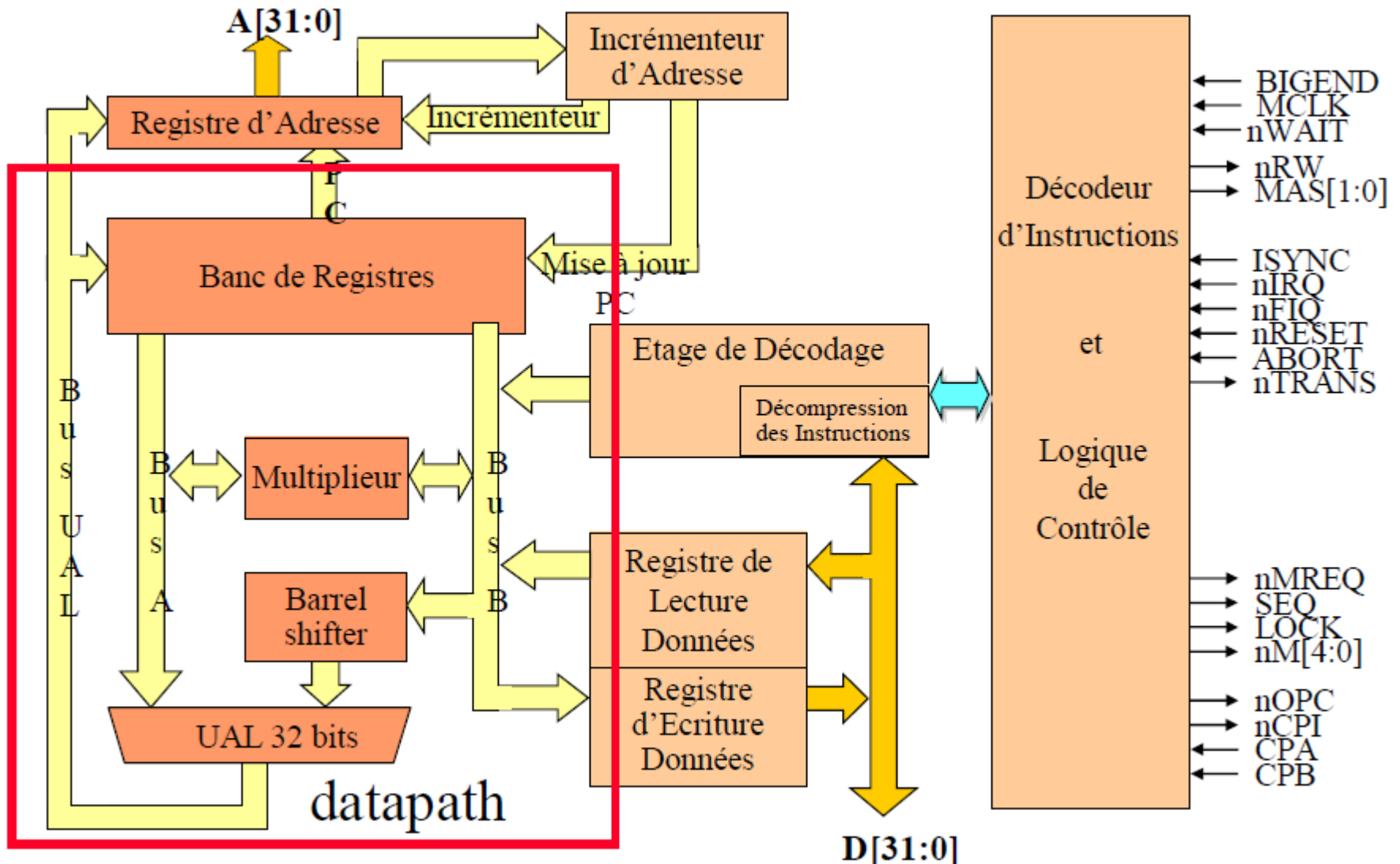
Vue d'une puce utilisant un ARM7

ARM710 (année 2000)



- Taille de la puce : 2.9 mm^2 en $0.18\mu\text{m}$.
- Taille du coeur : $0,5 \text{ mm}^2$ en $0.18\mu\text{m}$.

Vue simplifiée du cœur ARM7TDMI



Vue simplifiée du cœur ARM7TDMI

- L'architecture de l'ARM7TDMI est de type Von Neumann donc il n'y a qu'un bus pour véhiculer les instructions et les données.
- L'ARM dispose de 16 registres visibles sur lesquels les opérations sont effectuées.
- Les instructions LOAD/STORE servent respectivement à charger/enregistrer les données de/vers la mémoire.

Vue simplifiée du cœur ARM7TDMI

- La partie opérative (datapath) est constituée de 2 bus A et B sur lesquels se font les opérations.
- Le flot de calcul est le suivant :

Banc de registres sur A et B => Barrelshifter sur B
=> UAL => Banc de registres
- Les opérations sur ce datapath durent 1 cycle.

Vue simplifiée du cœur ARM7TDMI

Exemple

- L'exécution de l'instruction **ADD r0,r1,r2** consiste à mettre r1 et r2 sur A et B, effectuer l'addition dans l'UAL et écrire le résultat dans r0.

$$(r0) := (r1) + (r2)$$

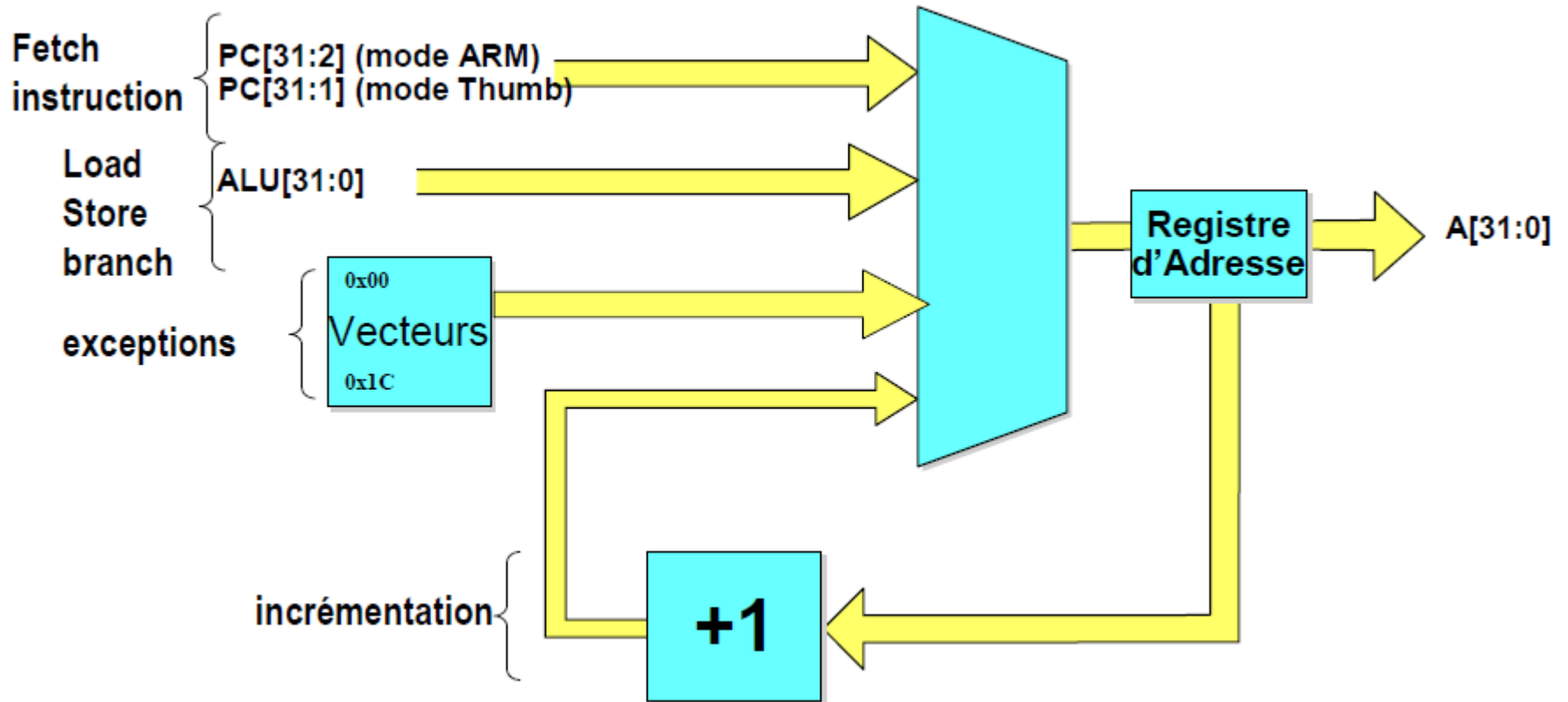
- Une opération de décalage aurait pu avoir lieu sur r2 dans le même cycle **ADD r0,r1,r2,LSL#2**. Ici r2 est décalé de 2 bits sur la gauche avant d'être additionné.

$$(r0) := (r1) + 4 * (r2)$$

Vue simplifiée du cœur ARM7TDMI

- Le registre d'adresse fournit les adresses à la mémoire.
- Les blocs décompression et décodage des instructions servent aux 2 premiers cycles du pipeline alors que le datapath sert au 3ème cycle.
- Les registres de Lecture et d'écriture données sont des tampons de données avec la mémoire externe pour les instructions LOAD/STORE.
- Le bloc de contrôle à droite est l'interface avec tous les signaux de commande externes : contrôleur mémoire, coprocesseurs, interruptions...

Génération des adresses



Génération des adresses

L'adresse $A[31:0]$ fournie à la mémoire externe peut être générée par 4 sources différentes

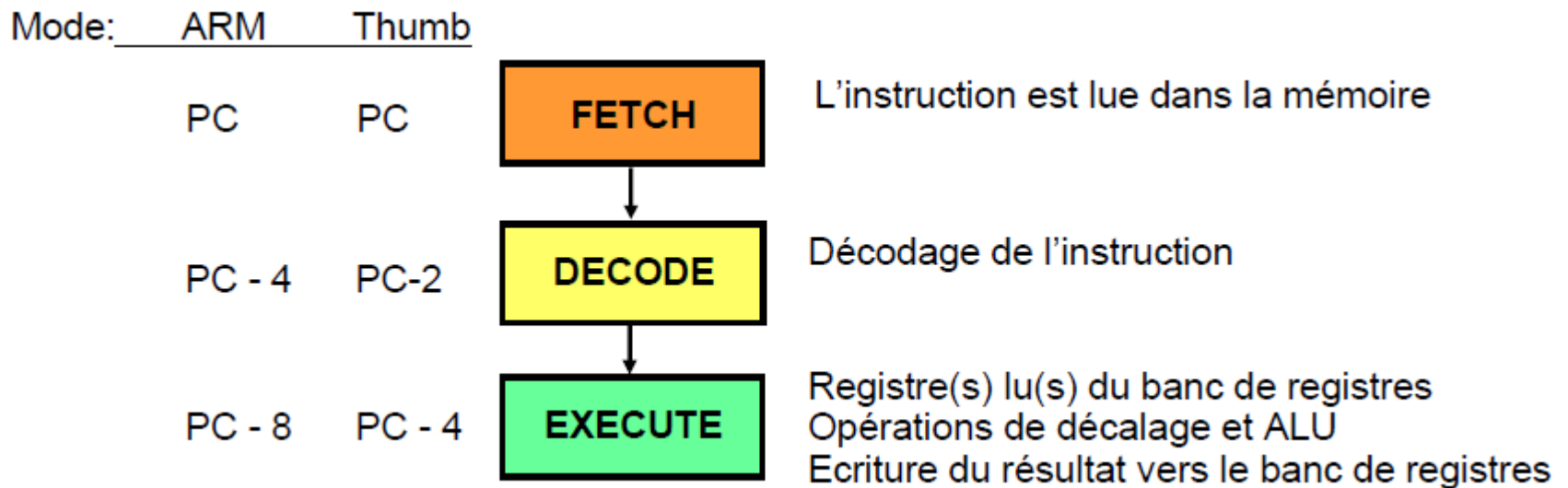
1. Le **registre d'adresse +1** , cas le plus fréquent où les instructions se suivent. Il correspond à la valeur du compteur de programme PC incrémenté de 1 de façon à aller chercher l'instruction suivante. Comme le chemin d'incrémentation avec le vrai registre PC est critique, il est plus efficace d'effectuer l'incrémentation avec le registre d'adresse si celui-ci est préalablement chargé avec PC.

Génération des adresses

2. Le compteur de programme **PC**. En fait il ne sert qu'après les **LOAD** et **STORE** pour recharger le registre d'adresse avec la valeur de PC. Notez que les 2 bits de faible poids sont inutiles en mode 32 bits ARM car les instructions sont toujours alignées sur des mots de 32 bits. En mode Thumb, seul le bit de faible poids est inutile.
3. L'**UAL** lors de l'exécution d'un **LOAD**, **STORE** ou **BRANCH**, de façon à pointer sur une donnée en mémoire (**LOAD** ou **STORE**) ou effectuer un branchement.
4. Les **vecteurs** sont des adresses pour pointer vers les programmes des traitements d'exceptions.

Le Pipeline d'instructions

La famille ARM7 utilise un pipeline à 3 étages pour augmenter la vitesse du flot d'instructions dans le microprocesseur.



Le Pipeline d'instructions

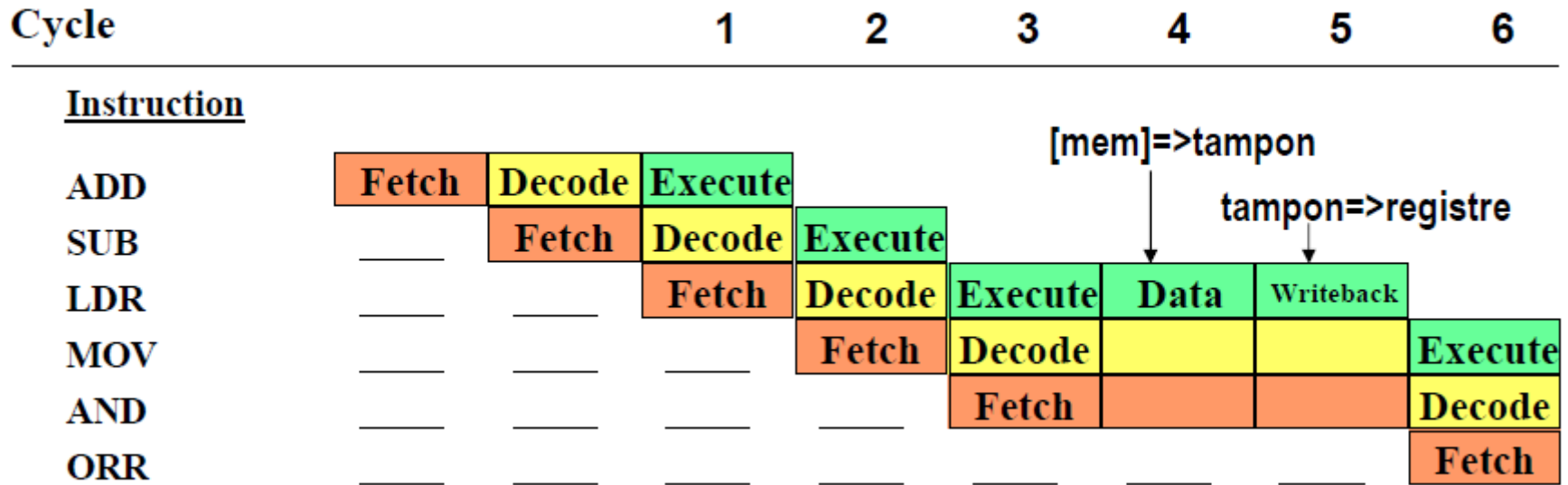
- Le PC pointe sur l'instruction en cours de lecture (FETCHed) et non sur l'instruction en cours d'exécution.
- Les instructions sont "pipelinées" en 3 cycles d'horloge : FETCH, DECODE et EXECUTE. Ainsi le concept RISC (1 instruction par cycle) est respecté.

Exemple 1 : Pipeline optimal

Cycle		1	2	3	4	5	6
<u>Instruction</u>							
ADD	Fetch	Decode	Execute				
SUB		Fetch	Decode	Execute			
MOV			Fetch	Decode	Execute		
AND				Fetch	Decode	Execute	
ORR					Fetch	Decode	Execute
EOR						Fetch	Decode
CMP							Fetch
RSB							

- Il faut 6 cycles pour exécuter 6 instructions (CPI “Cycles Per Instruction”=1).
- Toutes les instructions n'opèrent que sur des registres (1 cycle).

Exemple 2 : Pipeline avec LDR



- Le pipeline est rompu à cause de l'instruction LDR qui lit une donnée en mémoire et la charge dans un registre.
- Il faut 6 cycles pour exécuter 4 instructions. $CPI = 1,5$.
- Remarque : Les instructions LOAD et STORE prennent donc 3 cycles à la place de 1.

Les modes de l'ARM7TDMI

- Le processeur ARM a plusieurs modes de fonctionnement différenciés par des ressources et des privilèges spécifiques : certaines opérations ne sont autorisées que dans certains modes.
- Ces modes simplifient le portage d'un système d'exploitation multi-utilisateurs.
- Ils participent à la protection du code et des données ainsi qu'à la gestion d'erreurs.

Les modes de l'ARM7TDMI

Un ARM7TDMI a 7 modes opératoires de base

- **User** : sans privilège, la plupart des tâches s'exécutent
- **FIQ** : entrée par une interruption de priorité haute (Fast)
- **IRQ** : entrée par une interruption de priorité basse (normale)
- **Supervisor** : entrée à la réinitialisation et lors d'une interruption logicielle (SWI "SoftWare Interrupt")
- **Abort** : utilisé pour gérer les violations d'accès mémoire
- **Undef** : utilisé pour gérer les instructions non définies ("undefined")
- **System** : mode avec privilège utilisant les mêmes registres que le mode User.

Les registres

Registres actifs

Mode
Utilisateur

r0
r1
r2
r3
r4
r5
r6
r7
r8
r9
r10
r11
r12
r13 (sp)
r14 (lr)
r15 (pc)
cpsr

Bancs de Registres

(Spécifiques à un mode)

FIQ

IRQ

SVC

Undef

Abort

r8
r9
r10
r11
r12
r13 (sp)
r14 (lr)

r13 (sp)
r14 (lr)

r13 (sp)
r14 (lr)

r13 (sp)
r14 (lr)

r13 (sp)
r14 (lr)

spsr

spsr

spsr

spsr

spsr

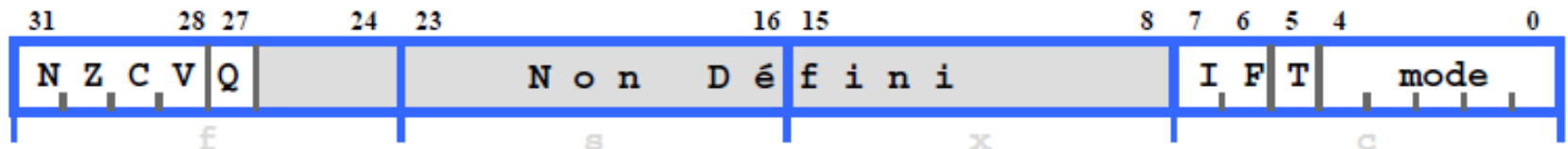
Les registres

- 37 registres organisés en bancs associés aux modes.
- 17 registres sont visibles :
 - 13 registres de données génériques (**r0 - r12**).
 - Le registre spécifique **r13** est réservé comme pointeur de pile et il est propre à chaque mode. Il s'appelle aussi **SP**.
 - Le registre spécifique **r14** est aussi le registre de retour de sous programme **LR**. Il évite l'empilage du PC et est propre à chaque mode.
 - Le registre spécifique **r15** est aussi le compteur ordinal **PC**.
 - Le registre de statut **CPSR** (Current Program Status Register) contient des informations sur l'état du processeur.

Les registres

- Le registre **SPSR** est une copie du CPSR avant de changer de mode. Ce registre évite l'empilage du CPSR et est propre à chaque mode.
- En mode "Fast Interrupt" FIQ, les registres de données sont nombreux de façon à ne pas avoir à sauvegarder le contexte des registres du programme principal et donc diminuer la latence pour accélérer le traitement.

Les registres d'état CPSR et SPSR



Validation des interruptions

- ☐ $I = 1$ dévalide IRQ.
- ☐ $F = 1$ dévalide FIQ.

Mode Thumb

- ☐ Architecture *xT* seulement
- ☐ $T = 0$, Processeur en mode ARM
- ☐ $T = 1$, Processeur en mode Thumb

Indicateurs de mode

- ☐ Indiquent le mode actif : système, IRQ, FIQ, utilisateur...

Indicateurs conditionnels

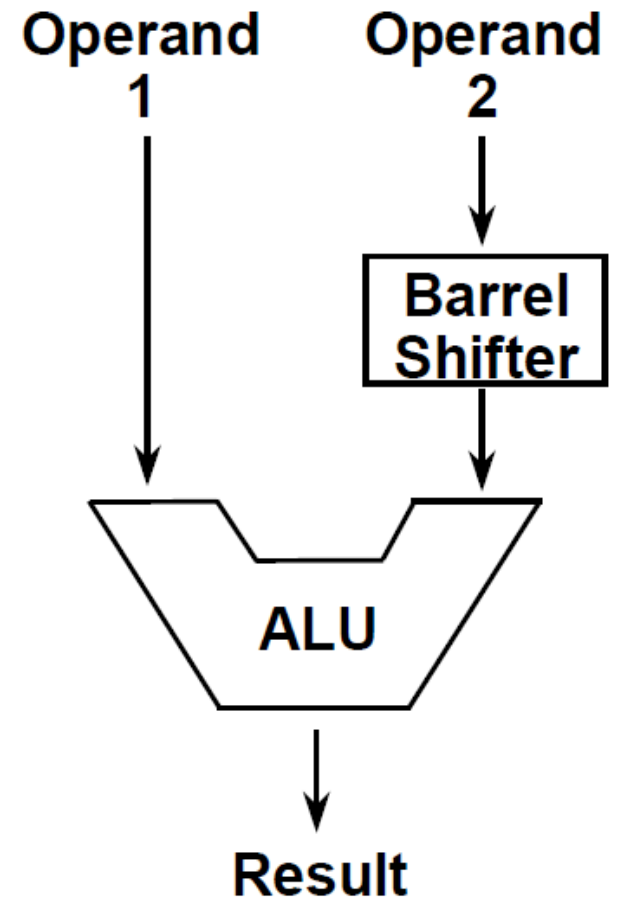
- ☐ N = Résultat Négatif de l'ALU
- ☐ Z = Résultat nul de l'ALU (Zéro)
- ☐ C = Retenue (Carry)
- ☐ V = Débordement (*o*Verflow)

Q = débordement avec mémoire

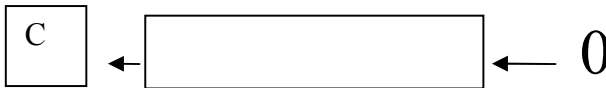
- ☐ Architecture 5TE seulement
- ☐ Indique qu'un débordement s'est produit pendant une série d'opérations

Le Barrel Shifter

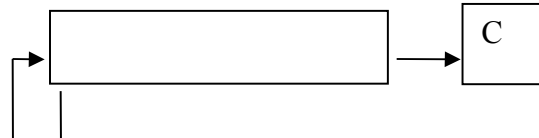
- L'ARM7 ne dispose pas d'instruction de décalages et de rotations de bits.
- A la place, il dispose d'un « Barrel Shifter » permettant d'effectuer des décalages et des rotations au sein d'autres instructions.



Opérations possibles avec le Barrel Shifter

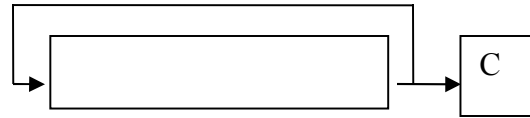
- Décalage logique à gauche (LSL) \equiv multiplication par une puissance de 2 : 

- Décalage logique à droite (LSR) \equiv division par une puissance de 2 : 

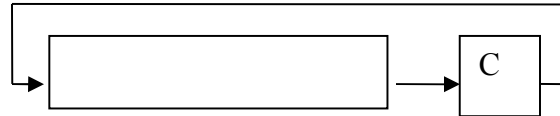
- Décalage arithmétique à droite (ASR) \equiv division par une puissance de 2 + préservation du signe pour le complément : 

Opérations possibles avec le Barrel Shifter

- La rotation à droite (ROR) :



- La rotation à droite étendue (RRX) :



Accès à la mémoire et aux périphériques

- 2 instructions d'accès seulement : LOAD (LDR) et STORE (STR).
- L'adressage mémoire se fait sur 32 bits. Il peut donc adresser 4 Go de mémoire.
- Les périphériques sont mappés en mémoire.
- Gestion mémoire « big endian » ou « little endian » (signal d'entrée CFCBIGEND).

Les exceptions

- Une exception est un évènement provoqué par une condition exceptionnelle lors de l'exécution de l'instruction courante :
 - Adresse mémoire invalide
 - Instruction non autorisée ou invalide
 - Division par zéro (faite par un coprocesseur)
- Les exceptions sont détectées par le CPU puis traitées dans le contexte du programme courant.

Les interruptions

Une demande d'interruption peut être

- déclenchée par un événement associé à un composant matériel :
 - début/fin d'entrée/sortie d'un périphérique
 - échéance de temps terminée
- un signal envoyé au processeur de l'extérieur par l'intermédiaire d'un PIC (Programmable Interrupt Controller).

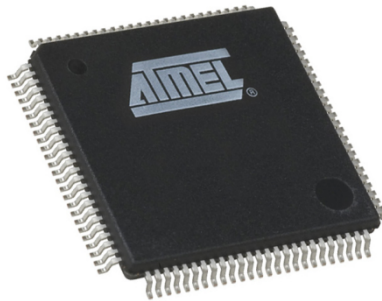
Traitement des exceptions et des interruptions

- Lorsqu'une exception ou une interruption survient :
 - le processeur suspend tout d'abord l'exécution en cours
 - sauvegarde ensuite l'état courant (PC, SP, registre d'état) dans des registres dédiés ou dans une pile
 - enfin charge le PC à une adresse mémoire spécifique qui dépend de l'exception ou de l'interruption
- Ces adresses sont localisées dans une partie de la mémoire appelée le vecteur d'exceptions / interruptions.

Vecteur d'exceptions / interruptions

- Le vecteur d'exceptions / interruptions contient généralement les adresses des fonctions permettant de traiter de manière appropriée les différents événements.
- Ces fonctions sont appelées gestionnaires d'exceptions ou d'interruptions.

Architecture de l'AT91SAM7S256



Généralités

- L'AT91SAM7S256 est un microcontrôleur fabriqué par la société ATMEL.
- Il est équipé d'un cœur de processeur ARM7TDMI.
- Sa fréquence de fonctionnement maximale est de 55 Mhz.
- Il dispose de 64 Ko de SRAM et de 256 Ko de mémoire FLASH.

Généralités

Caractéristiques de la mémoire Flash

- 1024 pages de 256 octets
- fréquence d'un cycle d'accès : 30 MHz
- 10 000 cycles d'écritures
- possibilité de rétention des données : 10 ans

Périphériques de l'AT91SAM7S256

- 1 port USB 2.0 ; 2 USART ; 1 port SPI ;
- 8 convertisseurs analogiques 10 bits ;
- 1 timer, 1 watchdog, 1 compteur ; 11 canaux DMA...

Schéma simplifié de l'AT91SAM7S256

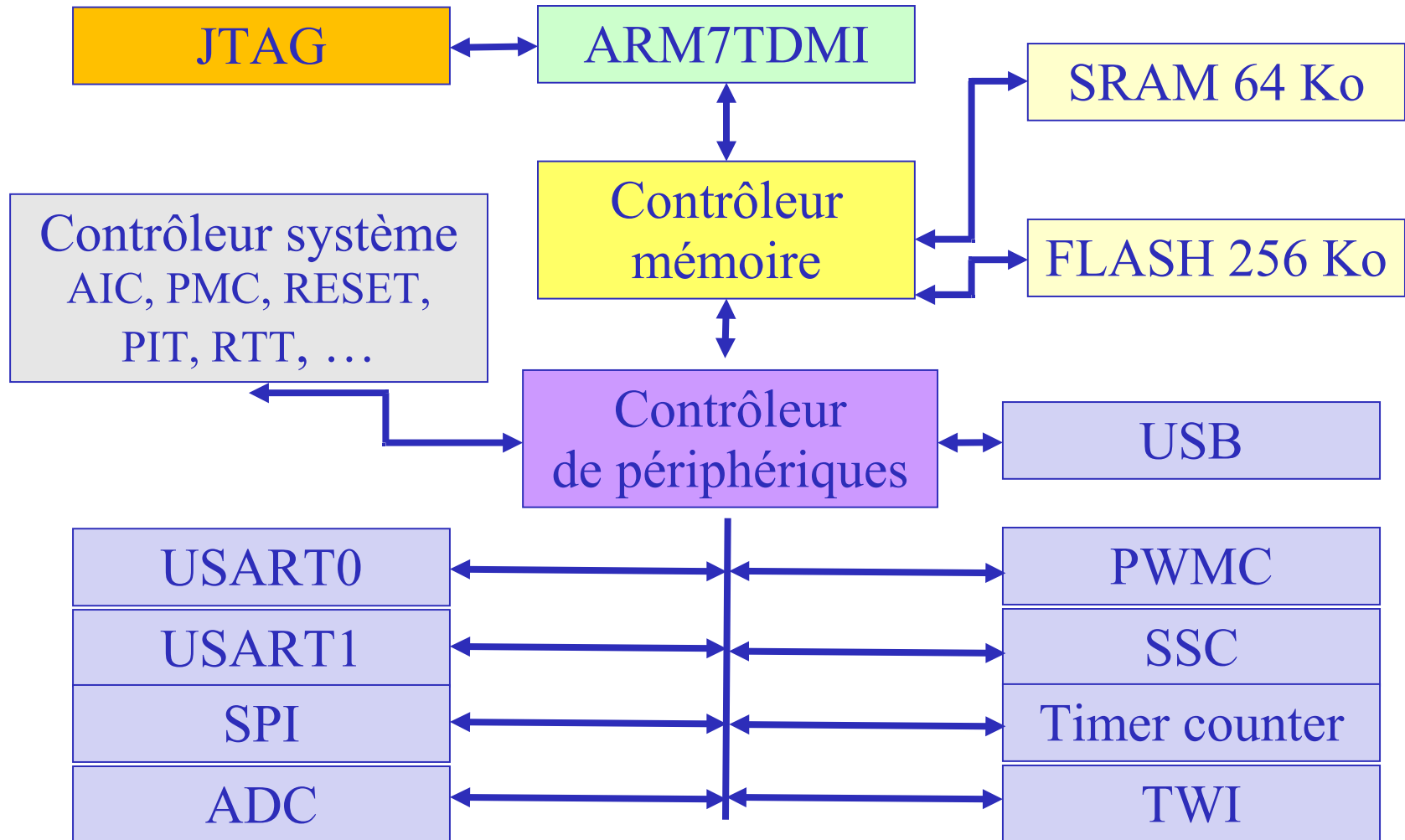
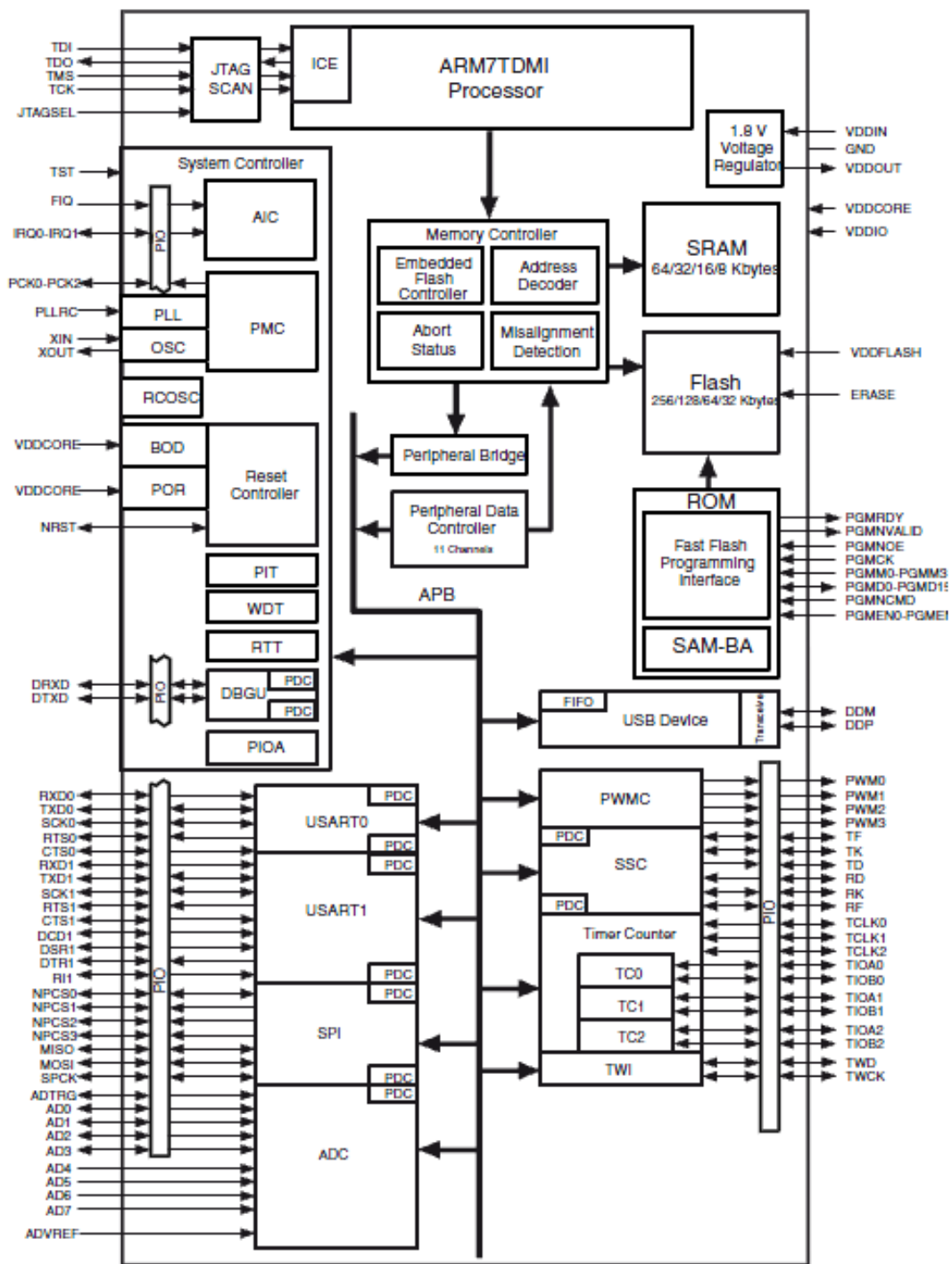


Schéma complet de l'AT91SAM7S256



Accès à la mémoire et aux périphériques

- Le processeur ARM a 32 bits d'adresse d'octets. Il adresse 4 Go de mémoire.
- Un **mot a une taille de 32 bits**. On peut accéder à des **demi-mots de 16 bits** et à des octets.
- Les mots doivent être alignés sur des adresses multiples de 4 et les demi-mots, de 2.
- Organisation de la mémoire par défaut : « little endian » petit bout d'abord (poids faibles).

Accès à la mémoire et aux périphériques

Les périphériques sont « mappés » dans la mémoire.

0x0000 0000 0x0FFF FFFF 0x1000 0000	Mémoires internes	256 Mo
0xEFFF FFFF 0xF000 0000	Plages d'adresses non utilisées	3584 Mo
0xFFFF FFFF	Périphériques	256 Mo

Adressage des mémoires internes

0x0000 0000	Flash (avant Remap) SRAM (après Remap)
0x000F FFFF	
0x0010 0000	Flash
0x001F FFFF	
0x0020 0000	SRAM
0x002F FFFF	
0x0030 0000	Plage d'adresses non utilisées
0x0FFF FFFF	

Adressage des périphériques embarqués

0xF000 0000

0xFFFFE FFFF

0xFFFF 0000

0xFFFF FFFF

Les périphériques
Le contrôleur système

Adressage des périphériques

	Peripheral Name	Size
0xF000 0000	Reserved	
0xFFFF 9 FFFF 0xFFFFA 0000	TC0, TC1, TC2	16 Kbytes
0xFFFFA 3FFF 0xFFFFA 4000	Reserved	
0xFFFFA FFFF 0xFFFFB 0000	UDP	16 Kbytes
0xFFFFB 3FFF 0xFFFFB 4000	Reserved	
0xFFFFB 7FFF 0xFFFFB 8000	TWI	16 Kbytes
0xFFFFB BFFF 0xFFFFB C000	Reserved	
0xFFFFB FFFF 0xFFFFC 0000	USART0	16 Kbytes
0xFFFFC 3FFF 0xFFFFC 4000	USART1	16 Kbytes
0xFFFFC 7FFF 0xFFFFC 8000	Reserved	
0xFFFFC BFFF 0xFFFFC C000	PWMC	16 Kbytes
0xFFFFC FFFF 0xFFFFD 0000	Reserved	
0xFFFFD 3FFF 0xFFFFD 4000	SSC	16 Kbytes
0xFFFFD 7FFF 0xFFFFD 8000	ADC	16 Kbytes
0xFFFFD BFFF 0xFFFFD C000	Reserved	
0xFFFFD FFFF 0xFFFFE 0000	SPI	16 Kbytes
0xFFFFE 3FFF 0xFFFFE 4000	Reserved	
0xFFFFE FFFF		

Adressage du contrôleur système

Address	Peripheral	Peripheral Name	Size
0xFFFF F000	AIC	Advanced Interrupt Controller	512 Bytes/128 registers
0xFFFF F1FF 0xFFFF F200			
0xFFFF F3FF 0xFFFF F400	DBGU	Debug Unit	512 Bytes/128 registers
0xFFFF F5FF 0xFFFF F600	PIOA	PIO Controller A	512 Bytes/128 registers
	Reserved		
0xFFFF FBFF 0xFFFF FC00	PMC	Power Management Controller	256 Bytes/64 registers
0xFFFF FCFF 0xFFFF FD00 0xFFFF FD0F	RSTC	Reset Controller	16 Bytes/4 registers
	Reserved		
0xFFFF FD20 0xFFFF FC2F	RTT	Real-time Timer	16 Bytes/4 registers
0xFFFF FD30 0xFFFF FC3F	PIT	Periodic Interval Timer	16 Bytes/4 registers
0xFFFF FD40 0xFFFF FD4F	WDT	Watchdog Timer	16 Bytes/4 registers
	Reserved		
0xFFFF FD60 0xFFFF FC6F	VREG	Voltage Regulator Mode Controller	4 Bytes/1 register
0xFFFF FD70 0xFFFF FEFF	Reserved		
0xFFFF FF00	MC	Memory Controller	256 Bytes/64 registers
0xFFFF FFFF			

Vecteur d'exceptions / interruptions

Vecteur d'interruption
de l' AT91SAM7S256
Fonction et
adresse de l'entrée de table

Reset	0X00000000
Undefined Instruction	0X00000004
Software Interrupt	0X00000008
Prefetch Abort	0X0000000C
Data Abort	0X00000010
Reserved	0X00000014
Interrupt request	0X00000018
Fast Interrupt Request	0X0000001C

IAR KickStart Kit

Généralités

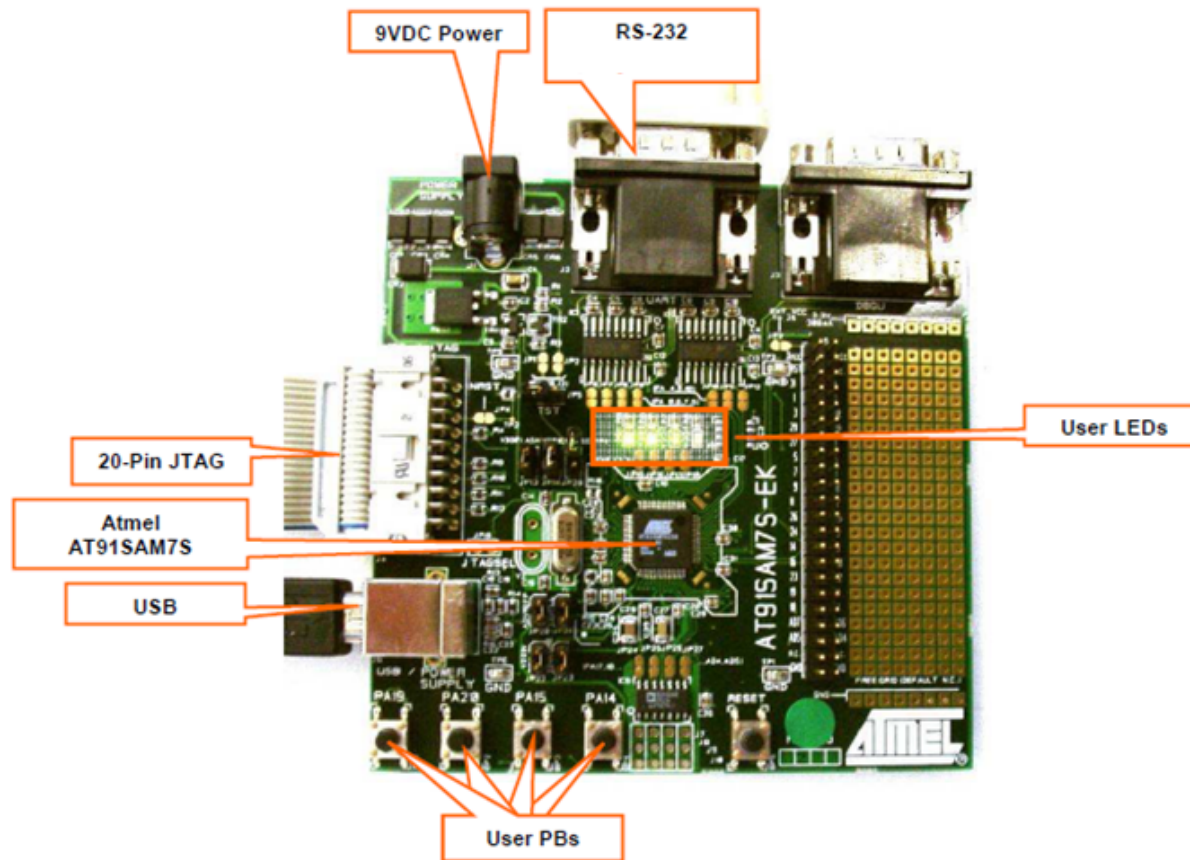
Le kit démarrage IAR pour AT91SAM7S256 regroupe

- la chaîne de développement Embedded Workbench
- une carte de développement AT91SAM7S-EK
- une sonde JTAG



La carte de développement AT91SAM7S-EK

Permet d'exécuter les programmes



La sonde JLINK

Interface permettant de charger un programme sur la carte de développement et de le déboguer : exécution pas à pas, points d'arrêt, suivi des variables, suivi des activités du processeur...

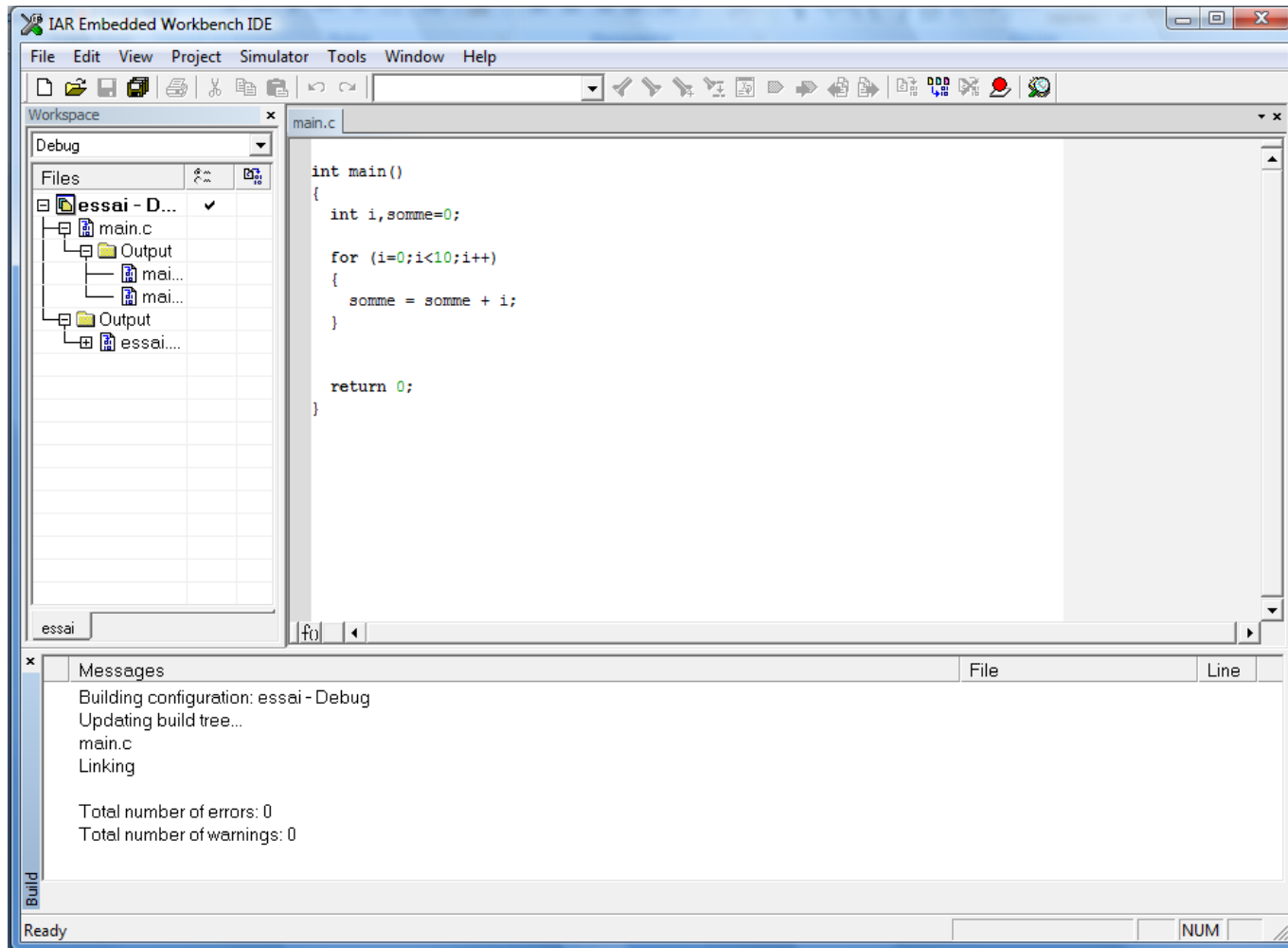


La chaîne de développement Embedded Workbench

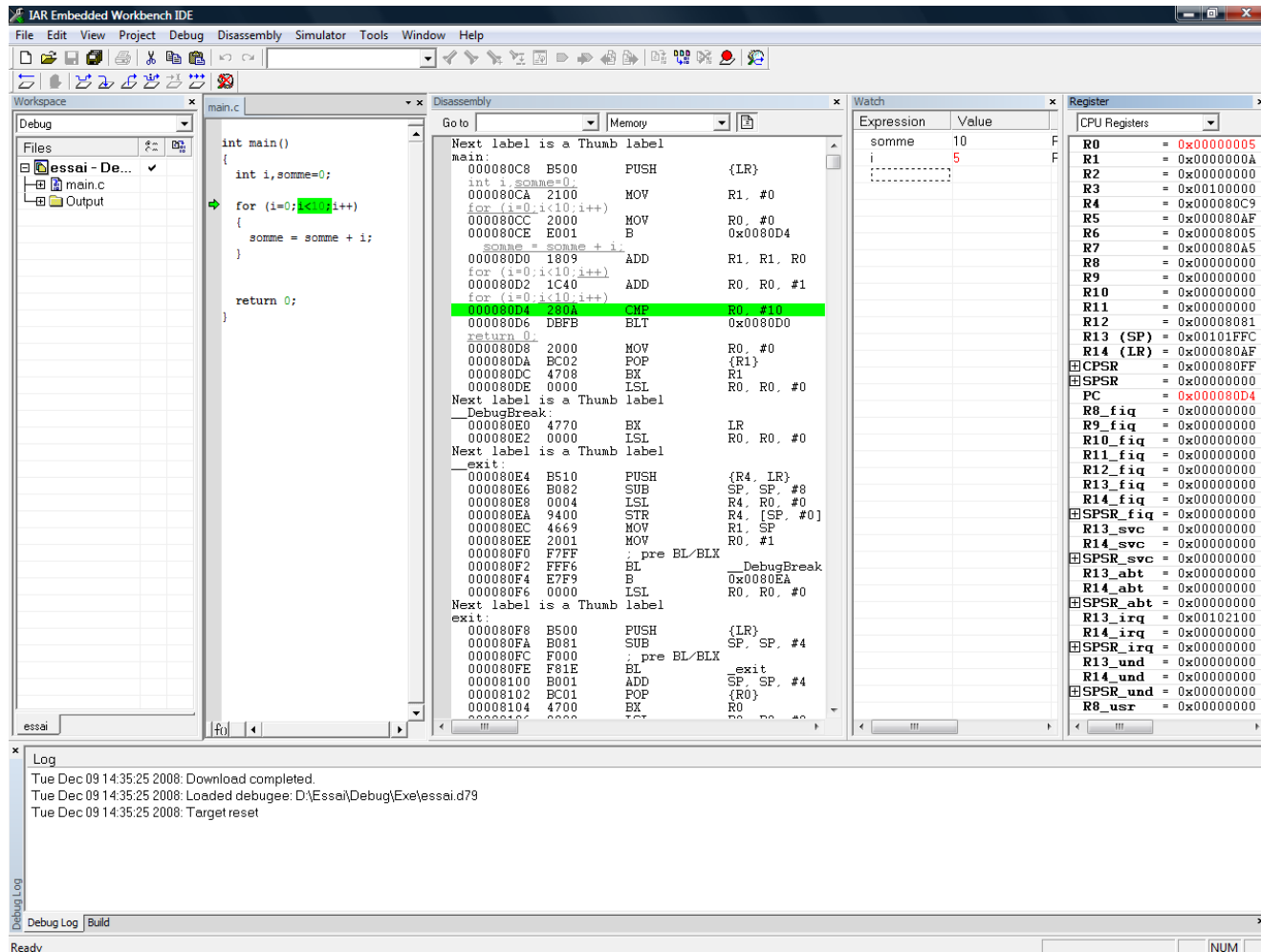
Environnement de développement intégré
assembleur / C / C++ comprenant

- un éditeur de code
- un gestionnaire de projet
- un compilateur
- un débogueur
- un simulateur

La chaîne de développement Embedded Workbench



La chaîne de développement Embedded Workbench

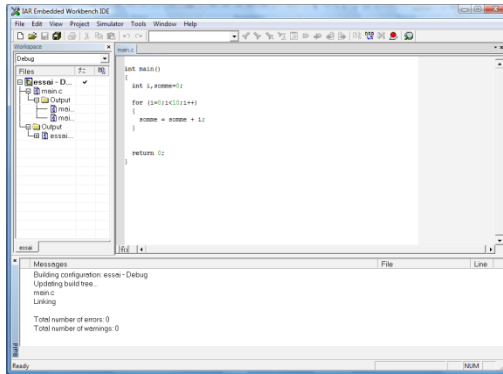


La chaîne de développement Embedded Workbench

Remarque

- La version d'Embedded Workbench livrée avec le kit de développement est une version limitée : les programmes ne peuvent pas dépasser 32 Ko.
- Cependant cette version est gratuite. Elle peut être téléchargée, après inscription en ligne, à l'adresse :
<http://supp.iar.com/Download/SW/?item=EWARM-KS32>

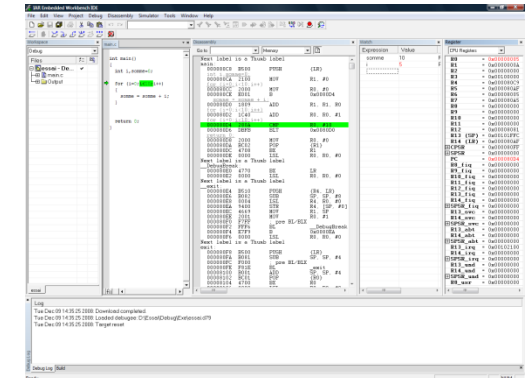
Processus de développement



1- Ecrire et compiler
le code



2- Le charger sur la carte



3- L'exécuter et
le déboguer

