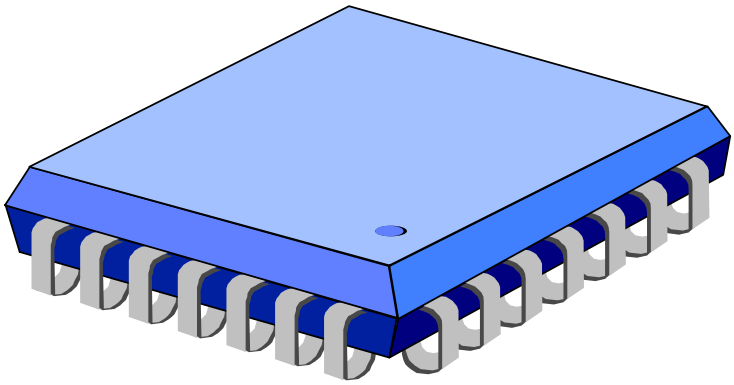
ARCHITECTURE de l'ARM7TDMI



ELEC 240
J-L Danger

TELECOM PARIS
école nationale supérieure des télécommunications

Qu’est ce qu’un ARM7TDMI?

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

TELECOM PARIS
école nationale supérieure des télécommunications

Le processeur **ARM7** est un processeur RISC 32 bits très utilisé dans les téléphones portables. Il s’agit en fait d’un « cœur » de processeur initialement conçu par la société britannique ACORN. Un "cœur" signifie que ce processeur est vendu comme bloc à utiliser dans un circuit qui intègre d'autre blocs pour constituer un système sur puce "SoC". Son succès vient de sa petite taille et sa faible consommation. Il représente un très bon exemple pédagogique de par son architecture RISC simple et classique.

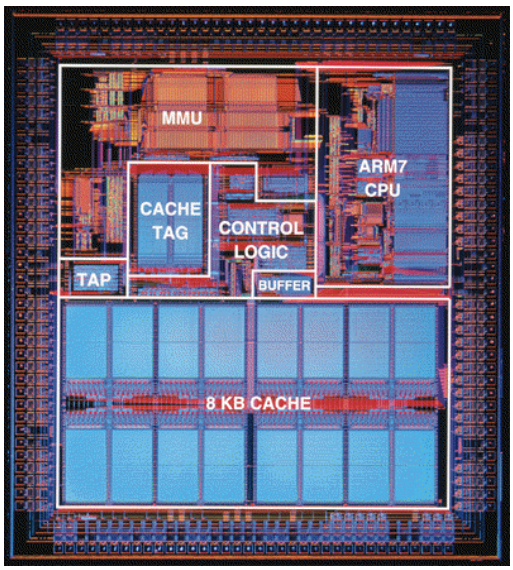
La lettre **T** symbolise le mode "Thumb" qui permet d’utiliser des instructions de 16 bits plutôt que 32 bits, de façon à diminuer la taille de la mémoire et donc le coût des équipements.

Le **D** signifie "Débugging" car le processeur dispose des facilités de débogage avec un "scan chain" autour du cœur de façon à pouvoir accéder aux bus données, adresse et contrôle du processeur.

Le **M** indique que le processeur dispose d’un multiplieur 32x8 lui permettant d’accélérer quelque peu les calculs de traitement du signal

Le **I** veut dire "In circuit Emulator" car le processeur a de la logique peu intrusive qui permet au concepteur d'analyser et piloter le déroulement de programme pour trouver les erreurs ou mettre au point l'environnement du processeur. L'émulation vient du fait qu'il est possible de piloter le processeur par l'extérieur et de lui adjoindre des ressources supplémentaires.

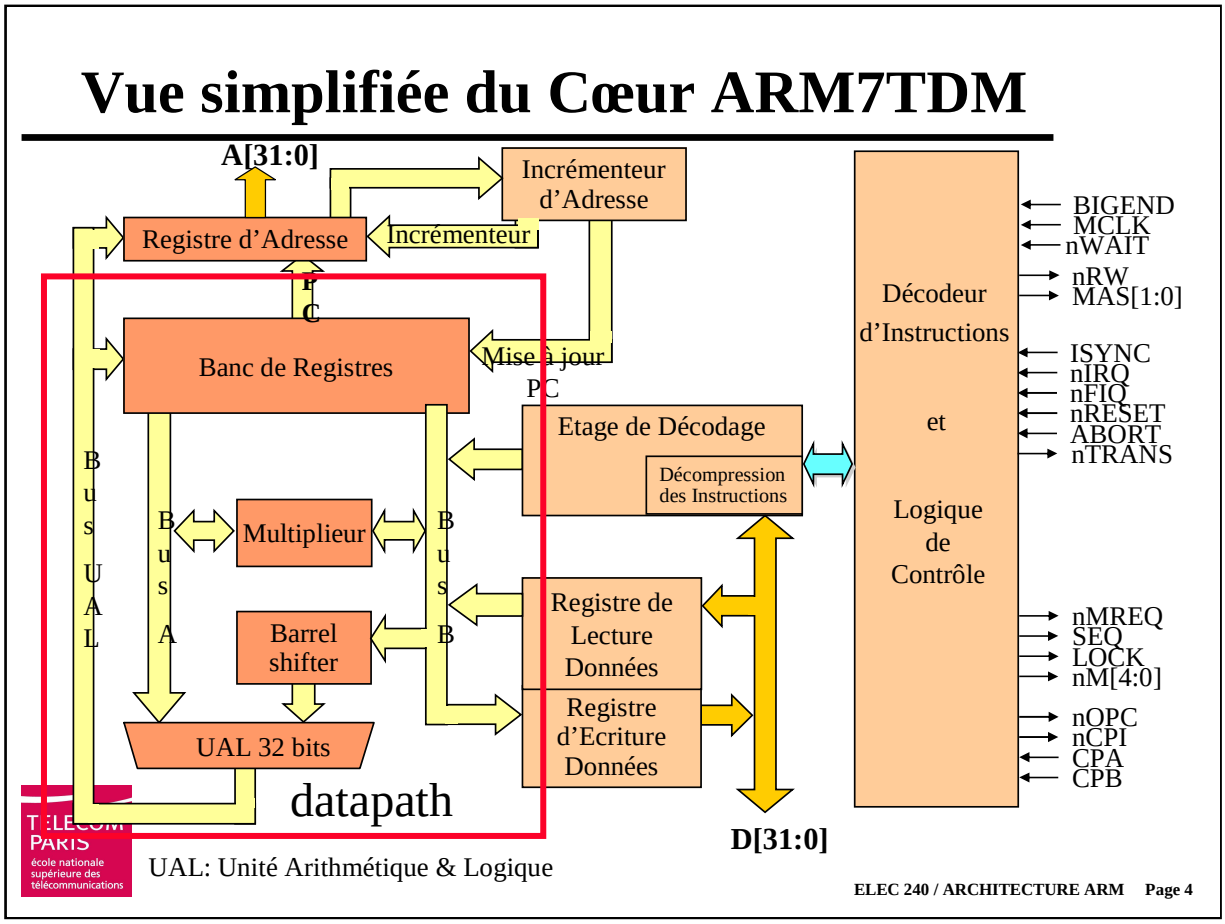
Vue d'une puce utilisant un ARM7



ARM710 (25mm² en 0.5µm (1995), 2.9 mm² en 0.18µm (2000))

TELECOM PARIS
école nationale supérieure des télécommunications

Voici un exemple d'un circuit utilisant le cœur ARM7. La taille du processeur ($0{,}5mm^2$ en 180nm) est relativement petite par rapport à la mémoire cache et au circuit de gestion mémoire MMU.



L'architecture de l'ARM 7 est Von Neuman, c'est-à-dire qu'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 précharger/stocker les résultats de/vers la mémoire.

La partie opérative (le "datapath") est constitué 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. L' exécution de l'instruction

ADD r0,r1,r2

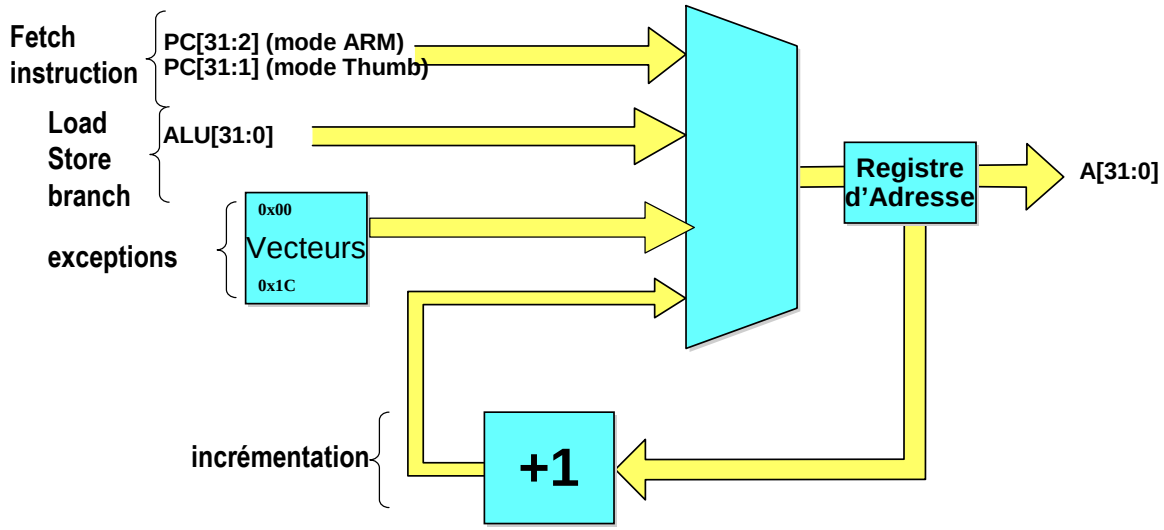
consiste à mettre r1 et r2 sur A et B, a effectuer l'addition dans l'UAL et écrire le résultat dans r0. Une opération de décalage aurait pu avoir lieu sur r2 dans le même cycle

ADD r0,r1,r2,LSL#2

Ici r2 est décalé de 2 bits sur la gauche avant d'être additionné

Le registre d'adresse fournit les adresses à la mémoire. Les blocs décompression et décodage des instructions servent au 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, comme par exemple les interruptions, le contrôleur mémoire, les coprocesseurs,…

Génération des Adresses



TELECOM PARIS
école nationale supérieure des télécommunications

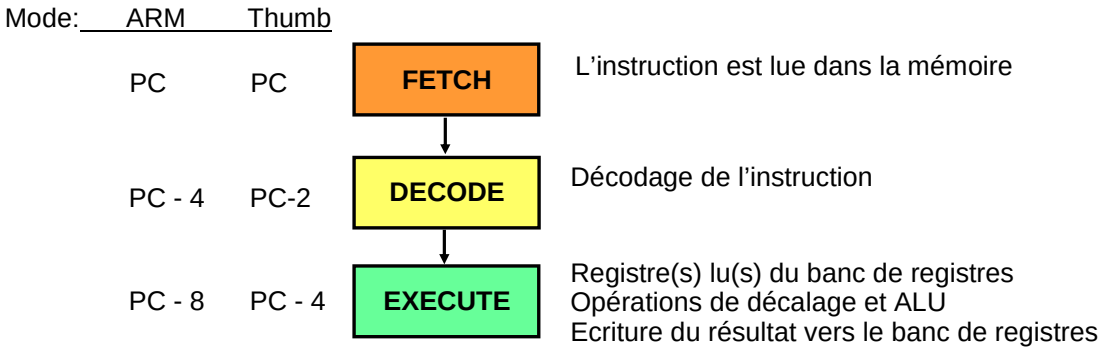
ELEC 240 / ARCHITECTURE ARM Page 5

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

1. Le **registre d'adresse +1**. C'est le cas le plus fréquent correspondant au cas 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.
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** qui sont des adresses pour aller 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.

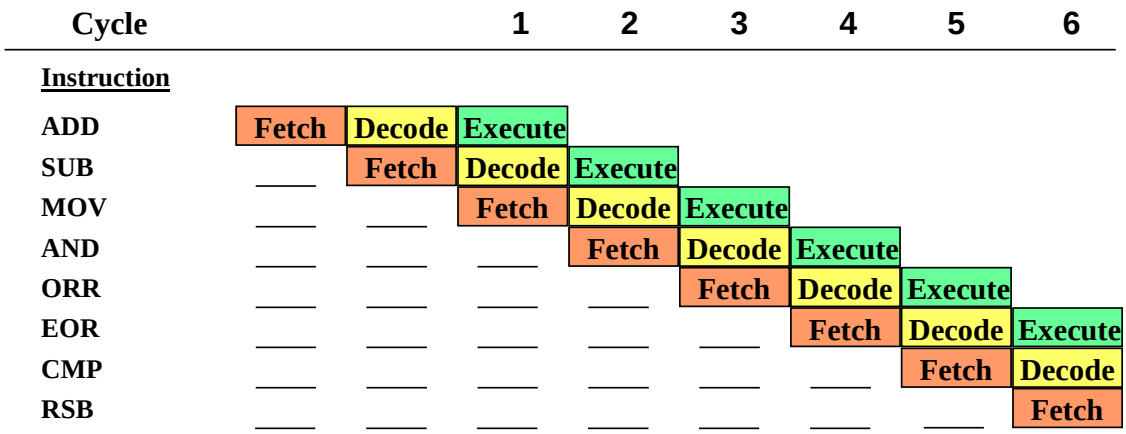


- Le PC pointe sur l'instruction en cours de lecture (FETCHed), et non sur l'instruction en cours d'exécution.

TELECOM PARIS école nationale supérieure des télécommunications

Les instructions sont "pipelinées" en 3 cycles d'horloge : FETCH, DECODE et EXECUTE. Ainsi le concept RISC (1 instruction par cycle) est respecté.

Exemple: Pipeline Optimal

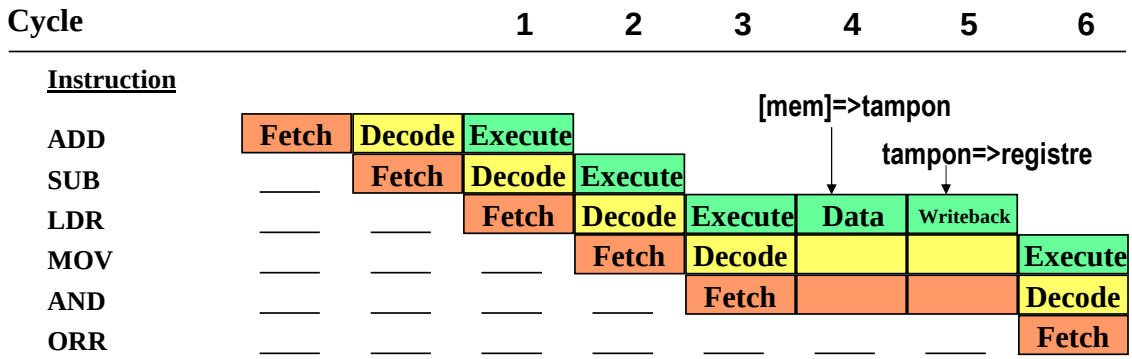


- il faut 6 cycles pour exécuter 6 instructions (CPI “Cycles Per Instruction”)=1
- Toutes les operations ne jouent que sur des registres (1 cycle)

TELECOM PARIS
école nationale supérieure des télécommunications

Voici le pipeline idéal avec un débit de calcul de 1 cycle par instruction (CPI=1)

Exemple: Pipeline avec LDR



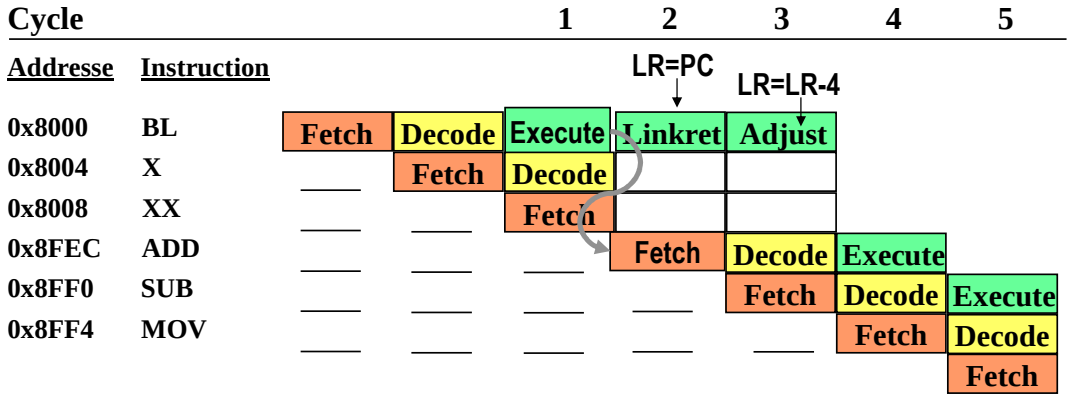
- L'instruction LDR lit une donnée en mémoire et la charge dans un registre
- il faut 6 cycles pour exécuter 4 instructions. CPI = 1,5

TELECOM PARIS
école nationale supérieure des télécommunications

Le pipeline est rompu lors d'un LOAD ou STORE. Dans cet exemple le LDR (LOAD Register) nécessité d'aller chercher la donnée dans la phase d'exécution. 1 cycle (cycle 4) est nécessaire pour lire la mémoire et mettre la donnée dan un registre tampon. Un autre cycle (cycle 5) sert à transférer la donnée du registre tampon vers le banc de registre.

Les instructions LOAD et STORE prennent donc 3 cycles à la place de 1.

Exemple: Pipeline avec Saut



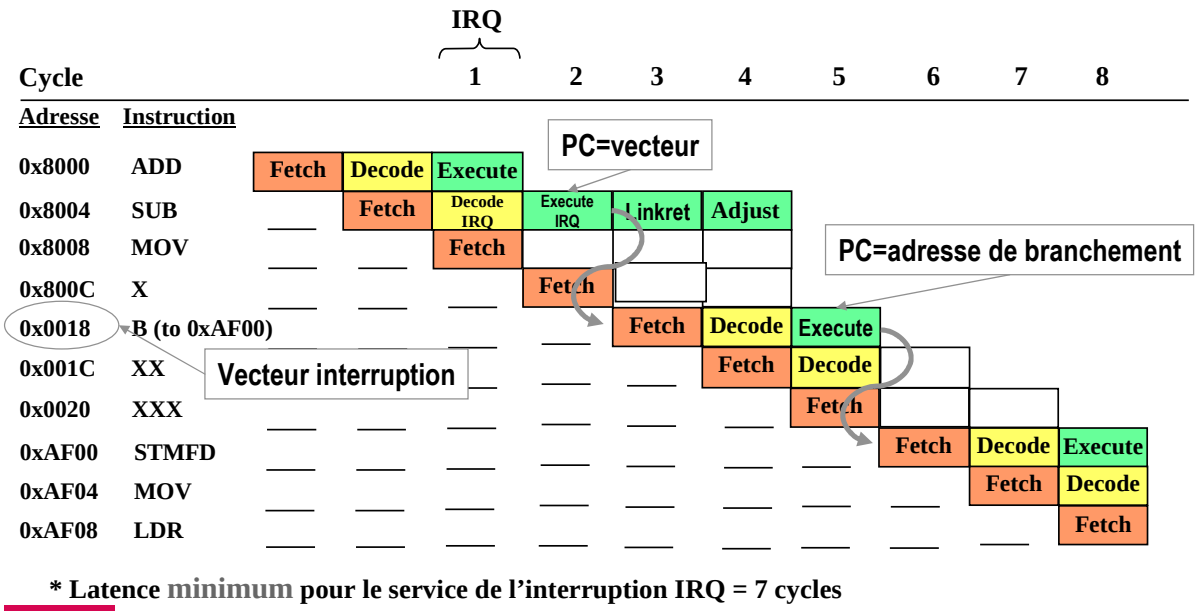
- L'instruction BL effectue un appel de sous-programme
- Le pipeline est cassé et 2 cycles sont perdus

TELECOM PARIS
école nationale supérieure des télécommunications

L'instruction de branchement BRANCH est exécutée au cycle 1, c'est-à-dire que le PC est chargé à la nouvelle valeur. Dans cet exemple il s'agit plus précisément d'un branchement à un sous-programme **BL**, le L signifiant "Link"pour le retour au programme principal. Au cycle 2 la première instruction du sous-programme est lue (FETCH) et le datapath est utilisé pour sauver le PC dans le registre **LR** "Link Register". Au 3ème cycle la valeur du LR est ajustée de façon à ce que le sous-programme revienne précisément à l'instruction suivant le saut au sous-programme.

Le branchement prend donc 3 cycles à la place de 1 cycle.

Exemple: Pipeline avec Interruption



*** Latence minimum pour le service de l'interruption IRQ = 7 cycles**

TELECOM PARIS
école nationale supérieure des télécommunications

Le déroulement est le suivant lorsque le processeur reçoit une interruption IRQ :

Cycle1: Le processeur est averti de l'interruption et vérifie si l'interruption est masquée. Si c'est le cas le processeur n'est pas interrompu, sinon il commence son exécution au prochain cycle.

Cycle 2: Exécution, c'est-à-dire récupération du vecteur, 0x18 pour IRQ, changement de mode, transfert CPSR vers SPSR (voir plus loin la signification de ces registres).

Cycle 3: Sauvegarde de PC-4 dans LR , lecture de l'instruction du branchement au programme de traitement BRANCH 0xAF00.

Cycle 4 et 5: cycles perdus à cause du branchement

Cycle 6: Lecture de la première instruction du programme de traitement de l'exception.

Le programme commence généralement par la sauvegarde du registre ne pile (empilage) et se termine par la récupération (dépilage). Le retour au programme principal s'effectue avec SUBS PC, LR ,#4 car le LR n'est pas exactement la valeur de retour au programme principal.

Les Modes du Microprocesseur

- **Un microprocesseur ARM a 7 modes opératoires de base :**
  - *User* ***: mode sans privilège où la plupart des tâches s'exécutent***
  - *FIQ* ***: on y entre lors d'une interruption de priorité haute (rapide)***
  - *IRQ* ***: on y entre lors d'une interruption de priorité basse (normale)***
  - *Supervisor* ***: on y entre 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***

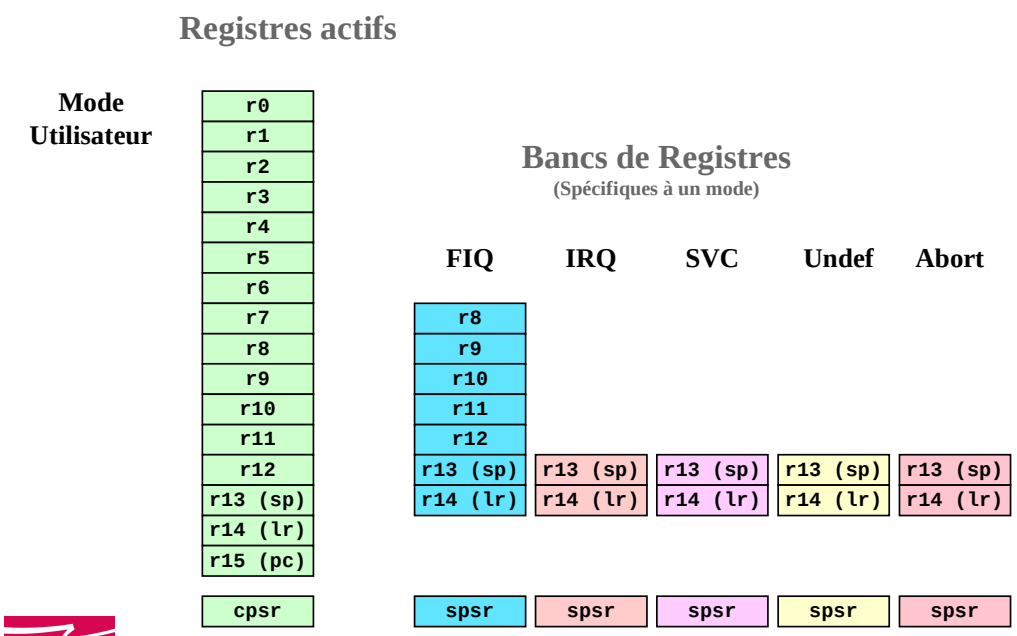
TELECOM PARIS
école nationale supérieure des télécommunications

Le processeur ARM a plusieurs modes de fonctionnement différenciés par des ressources et des privilèges spécifiques. Ces modes simplifient le portage d'un système d'exploitation multi-utilisateurs.

Selon les modes, le coeur ARM active des signaux de commandes : nTRANS, HPROT qui peuvent être utilisés par des contrôleurs mémoire ou E/S externes pour autoriser ou non certaines zones mémoire.

D'autre part certaines opérations ne sont autorisées que dans certains modes.

Les Registres



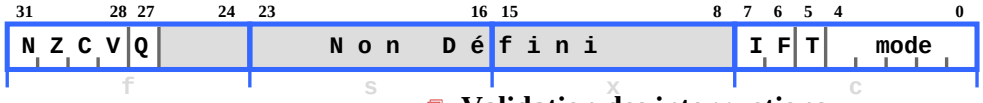
TELECOM PARIS
école nationale supérieure des télécommunications

L'architecture ARM fournit 37 registres organisés en bancs associés aux modes.17 registres sont visibles

- 13 registres de données génériques (r0 - r12).
- Le registre spécifique r13 qui est réservé comme pointeur de pile et est propre à chaque mode. Il s'appelle aussi **SP**.
- Le registre spécifique r14 qui est aussi le registre de retour de sous-programme **LR**. Ce registre évite l'empilage du PC et est propre à chaque mode.
- Le registre spécifique r15 qui est aussi le compteur de programme **PC**.
- Le registre de statut **CPSR** (Current Program Status Register) qui contient des informations sur l'état du processeur
- Le registre **SPSR** qui est une copié 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)



- **Indicateurs conditionnels**
  - *N = Résultat Négatif de l'ALU*
  - *Z = Résultat nul de l'ALU (Zéro)*
  - *C = Retenue (Carry)*
  - *V = Débordement (oVerflow)*
- **Q = débordement avec mémoire**
  - *Architecture 5TE seulement*
  - *Indique qu'un débordement s'est produit pendant une série d'opérations*
- **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...*

TELECOM PARIS
école nationale supérieure des télécommunications

Les bits de fort poids représentent les indicateurs de l'UAL. Les bits 6 et 7 sont des bits de contrôle pour masquer les interruptions. Le bit 5 permet de passer en mode Thumb et les 5 bits de faible poids indiquent le mode.

Accès à la Mémoire et aux E/S

- 2 instructions d'accès : LOAD (LDR) et STORE (STR)
- L'adressage mémoire se fait sur 32 bits, => 4 Go.
- Le type des données peut être: octets, demi-mots (16 bits) ou mots (32 bits)
- Les mots doivent être alignés sur des adresses multiples de 4 et les demi-mots, de 2.
- Les E/S sont dans la « mappe » mémoire

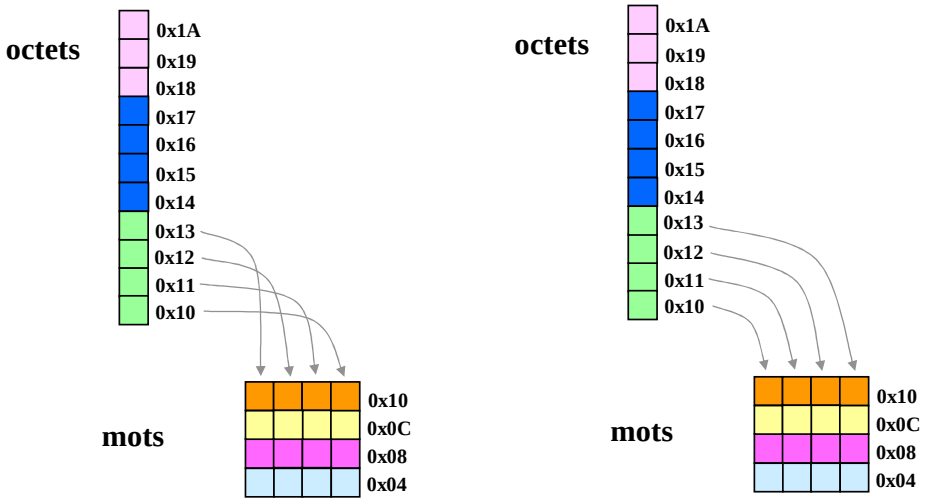
TELECOM PARIS
école nationale supérieure des télécommunications

Le processeur ARM dispose de 32 bits d'adresse octets pour accéder aussi bien à la mémoire qu'aux E/S. Un mot a par définition une taille de 32 bits. Il est possible d'accéder des demi-mots sur 16 bits et des octets.

Attention les mots ont toujours des adresses multiples de 4 (ils sont forcément alignés) et les demi-mots ont des adresses paires.

Organisation de la mémoire

La mémoire peut être vue comme une ligne d'octets repliée en mots.
2 façon d'organiser 4 octets en mot :



« Little Endian »

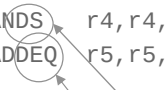
« Big Endian »

TELECOM PARIS
école nationale supérieure des télécommunications

Au sein d'un mot et d'un demi-mot, il y 2 façon d'adresser les octets. Ces 2 types d'adressage existent depuis les premiers microprocesseurs 16 bits où Intel (8086) et Motorola (68000) avaient choisi chacun un type différent.

Jeu d'Instructions ARM(1)

- Toutes les instructions ont 32 bits
- La plupart des instructions s'exécutent en un seul cycle
- Les instructions peuvent être exécutées conditionnellement
- Architecture Load/Store
  - ***Instructions de traitement de données***
    - `SUB  r0,r1,#5  ; r0= r1-5`
    - `ADD  r2,r3,r3,LSL #2  ; r2=R3+4*r3=5*r3`
    - `ANDS  r4,r4,#0x20  ; r4=r4 ET 0x20`
    - `ADDEQ  r5,r5,r6  ; r5=r5+r6 si Z`



Positionnement des indicateurs

Execution si le résultat précédent est 0

TELECOM PARIS
école nationale supérieure des télécommunications

Comme la majorité des langages "machine", la syntaxe du langage assembleur de l'ARM dispose de 5 champs pour décrire l'instruction :

champ1	champ2	champ3	champ4	champ5
Etiquette	Instruction	Résultat	Operande1	Opérande2

L'**étiquette** correspond à l'adresse symbolique. Elle est optionnelle à part pour les débuts de sous-programme.

L'**instruction** peut être complétée de la taille des données (8,16 ou 32 bits) et d'une condition d'exécution.

Le **résultat** est toujours avant les opérandes car certaines instructions n'ont qu'un seul opérande

Le résultat et les opérandes sont des indices de registre contenant les données

Les registres opérandes pour LOAD et STORE sont des registres d'adresse (adressage indirect)

L'opérande 2 peut être complétée par un décalage effectué par le "barrelshifter"

Jeu d'Instructions ARM(2)

- ***Instructions spécifiques d'accès à la mémoire***
  - `LDR r0,[r1],#4` *; r0=mem(r1), r1= r1+4*
  - `STRNEB r2,[r3,r4]` *; mem(r3+r4)=r2*
  - `LDRSH r5,[r6,#8]!` *; r5=mem(r6+8), r6=r6+8*
  - `STMFD sp!,{r0,r2-r4}` *; transferts multiples*

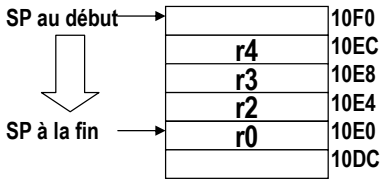
;empilage : mem(sp+i)=liste de registres

Si Z=0

En octet

En 16 bits
Avec extension
de signe sur les
16MSBs

r6=r6+8 en fin d'execution



Opération réciproque de dépilage :

```
LDMFD  sp!,{r0,r2-r4}
; liste de registres=mem(sp+i
```

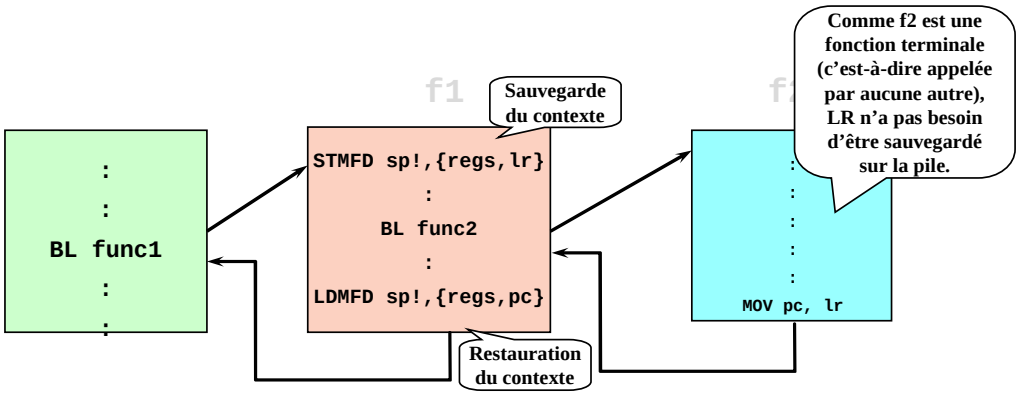
TELECOM PARIS
école nationale supérieure des télécommunications

Dans les instructions de transfert, le registre entre crochet [r1] indique qu'il s'agit d'un registre d'adresse servant à pointer une donnée en mémoire

Il existe beaucoup de déclinaisons des LOAD et STORE. Les LOAD et STORE multiples sont très utiles pour l'empilage ou le dépilage en utilisant le premier registre comme pointeur de pile (normalement c'est r13=SP)

Jeu d'Instructions ARM(3)

- Branchement et sous programmes :
  - `B <étiquette>`
    - ***Calculé par rapport au PC. Étendue du branchement: ±32 Mbyte.***
  - `BL <subroutine>`
    - ***Stocke l'adresse de retour dans le registre LR***
    - ***Le retour se fait en rechargeant le registre LR dans le PC***
    - ***Pour les fonctions non terminales, LR devra être sauvegardé***



TELECOM PARIS
école nationale supérieure des télécommunications

Le LR doit lui-même être empilé lorsqu'il y a des appels sous-programmes imbriqués. Remarquez l'instruction de dépilage à la fin du sous-programme f1 où un LOAD multiple permet de recharger le PC avec la valeur de LR.

Executions conditionnelles

- ***La plupart des instructions peuvent être exécutées conditionnellemnt aux indicateurs Z,C,V,N***

```
CMP       r0,#8    ; r0=8?
BEQ fin   ; si oui (Z=1) PC=fin
ADD       r1,r1,#4 ;

Équivalent à
CMP       r0,#8          ; r0=8?
ADDNE     r1,r1,#4     ; si non (Z=0)
```

+ petit et
+ rapide

Conditions courantes : EQ, NE,PL,MI,CS,VS

=0, ≠0,≥0,<0,carry set, debordement

TELECOM PARIS
école nationale supérieure des télécommunications

En mode 32 bits, les instructions exécutées conditionnellement permettent d'optimiser le code faisant appel à des branchements conditionnels.

Exemple: Séquence d'Instructions

```
LDR    R0, [R8, 0x10]
ADD    R1, R0, R4, LSL #2
STR    R1, [R8, 0x14]
```

charger le mot de l'adresse [R8+0x10] dans R0
R1 ← R0 + (R4 << 2)
ranger le contenu de R1 à l'adresse [R8+0x14]

- Conditions initiales :

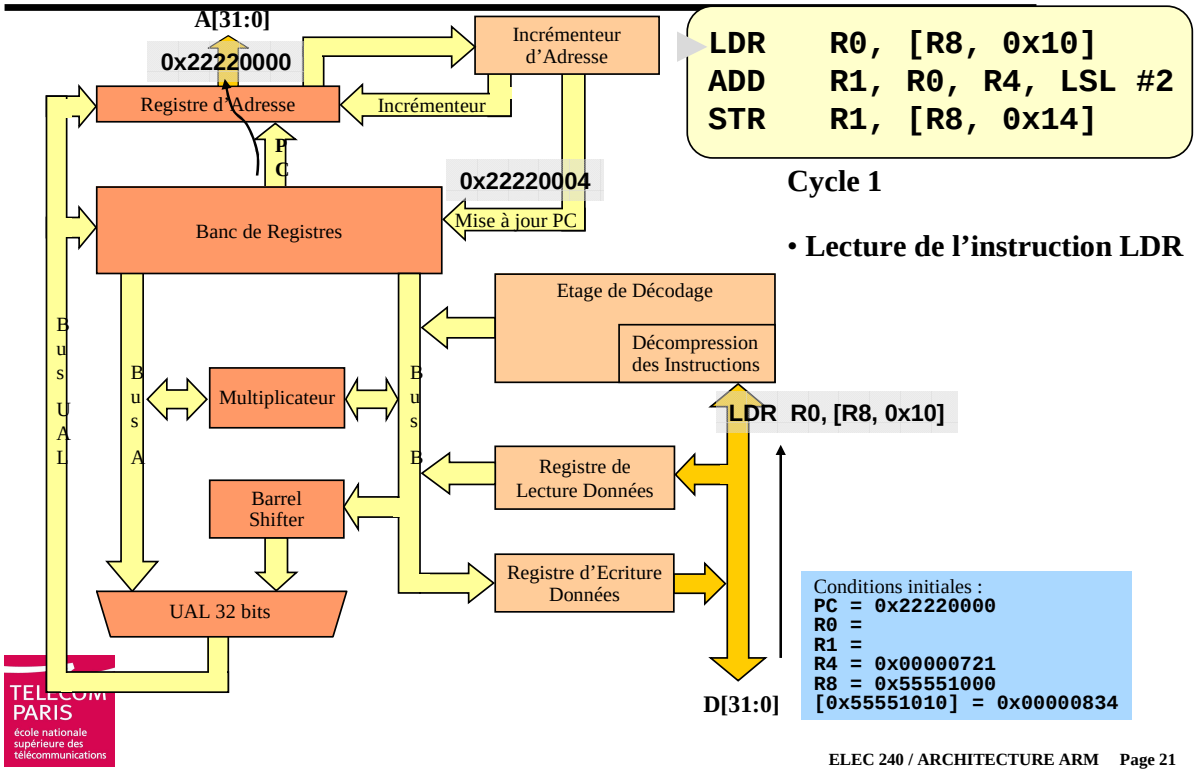
```
PC = 0x22220000
R4 = 0x00000721
R8 = 0x55551000
[0x55551010] = 0x00000834
```

- Les diagrammes suivants supposent que les instructions précédentes s'exécutent en un cycle mais ne montrent pas leur comportement.

TELECOM PARIS
école nationale supérieure des télécommunications

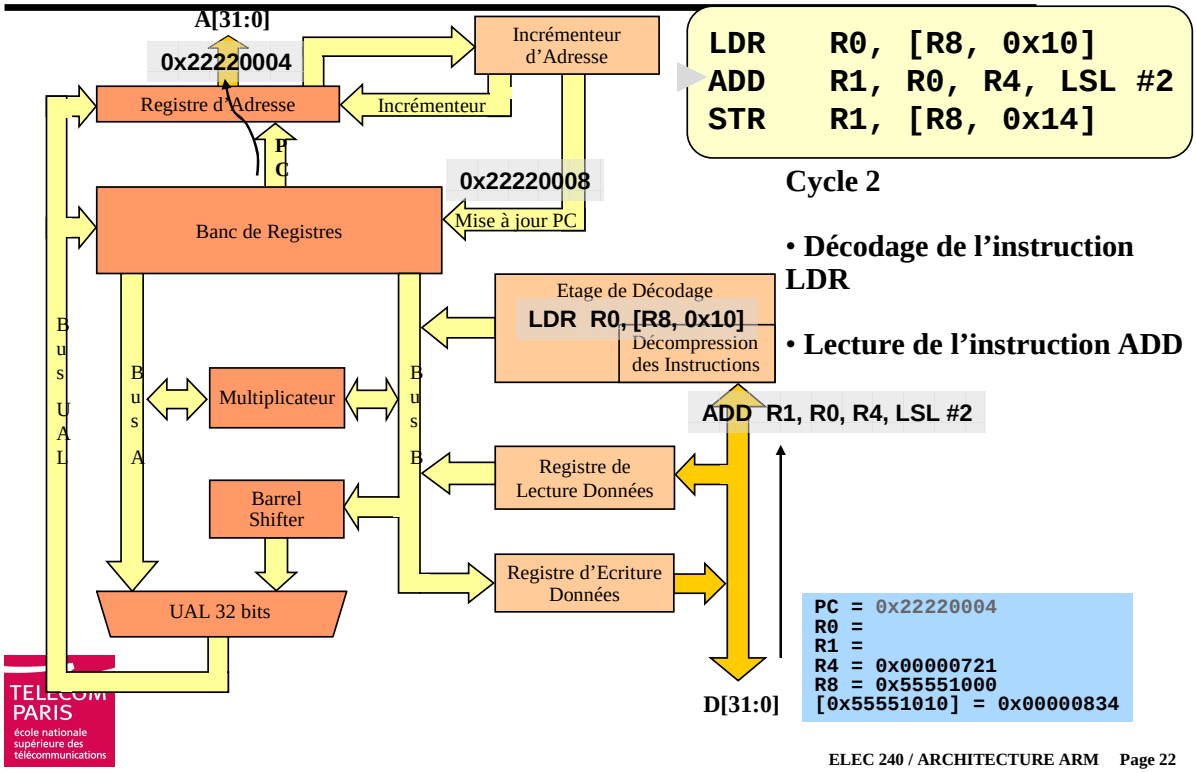
De façons à comprendre le fonctionnement interne de l'ARM, considérons ces 3 instructions avec cet état des registres.

Séquence d'Instructions : Cycle 1



L'instruction LDR est stockée dans un registre de l'étage de décompression.

Séquence d'Instructions : Cycle 2




```
LDR    R0, [R8, 0x10]
ADD    R1, R0, R4, LSL #2
STR    R1, [R8, 0x14]
```

Cycle 2

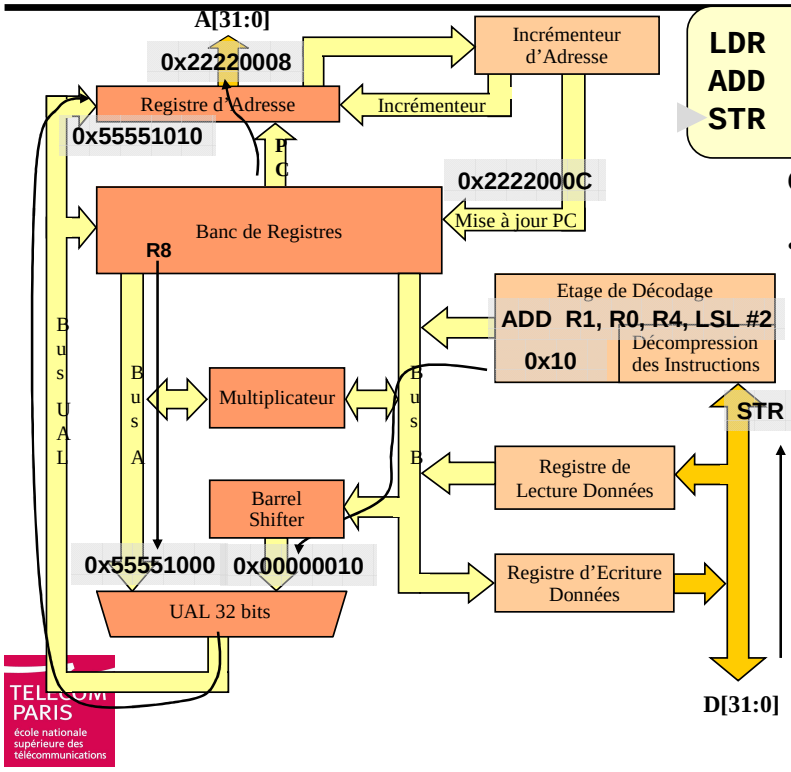
- **Décodage de l'instruction LDR**
- **Lecture de l'instruction ADD**

TELECOM PARIS
école nationale supérieure des télécommunications

LDR est décodée

ADD est lue dans le tampon d'instruction

Séquence d'Instructions : Cycle 3




```
LDR   R0, [R8, 0x10]
ADD   R1, R0, R4, LSL #2
STR   R1, [R8, 0x14]
```

Cycle 3

- **1er cycle d'exécution de LDR**
  - **Calcul de l'adresse en mémoire de données**
  - **Décodage de l'instruction ADD**

STR R1, [R8, #14]

- **Lecture de l'instruction STR**

```
PC = 0x2222000C
R0 =
R1 =
R4 = 0x00000721
R8 = 0x55551000
[0x55551010] = 0x00000834
```

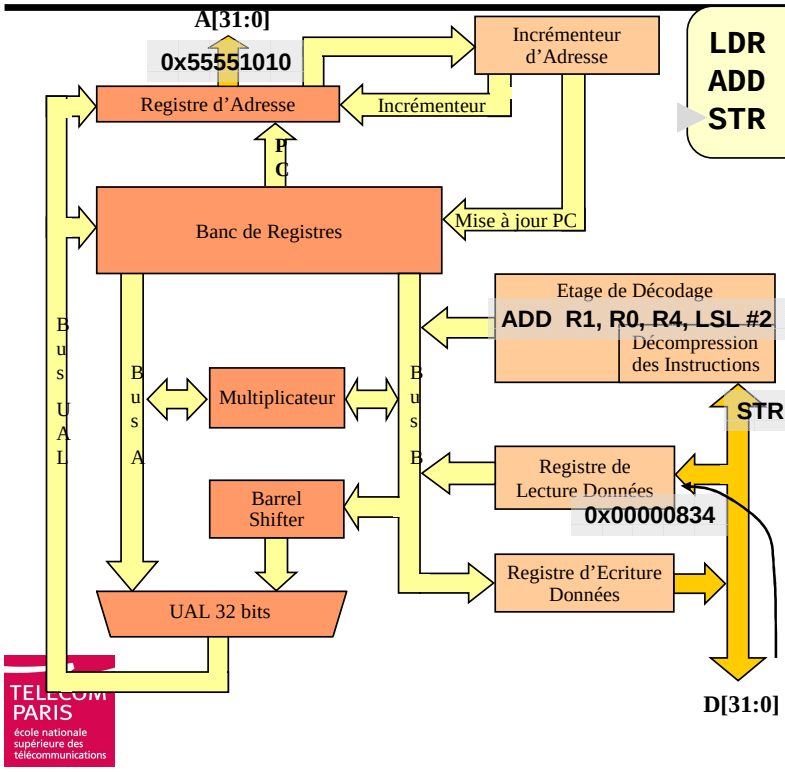
TELECOM PARIS école nationale supérieure des télécommunications

LDR est exécutée : Dans le datapath le calcul de R8 + 0x10 est effectué

ADD est décodée

STR est lue dans le tampon d'instructions

Séquence d'Instructions : Cycle 4




```
LDR    R0, [R8, 0x10]
ADD    R1, R0, R4, LSL #2
STR    R1, [R8, 0x14]
```

Cycle 4

- **2ème cycle d'exécution de LDR**
  - **Lecture de la mémoire de données**

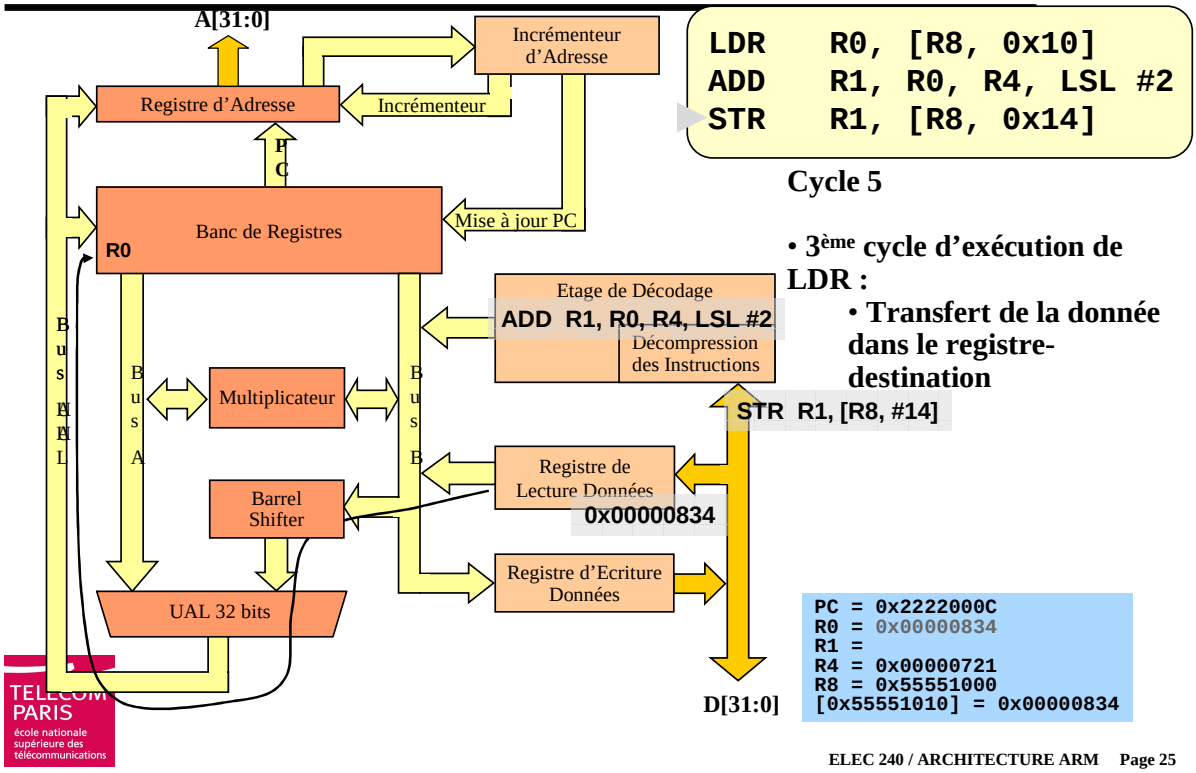
```
PC = 0x2222000C
R0 =
R1 =
R4 = 0x00000721
R8 = 0x55551000
[0x55551010] = 0x00000834
```

TELECOM PARIS
école nationale supérieure des télécommunications

Le LDR est toujours exécuté car il faut aller lire une donnée en mémoire avec l'adresse R8+0x10. Cette donnée est stockée dans le registre tampon de lecture données

Les ADD et STR sont gelés respectivement dans l'étage de décodage et de tampon.

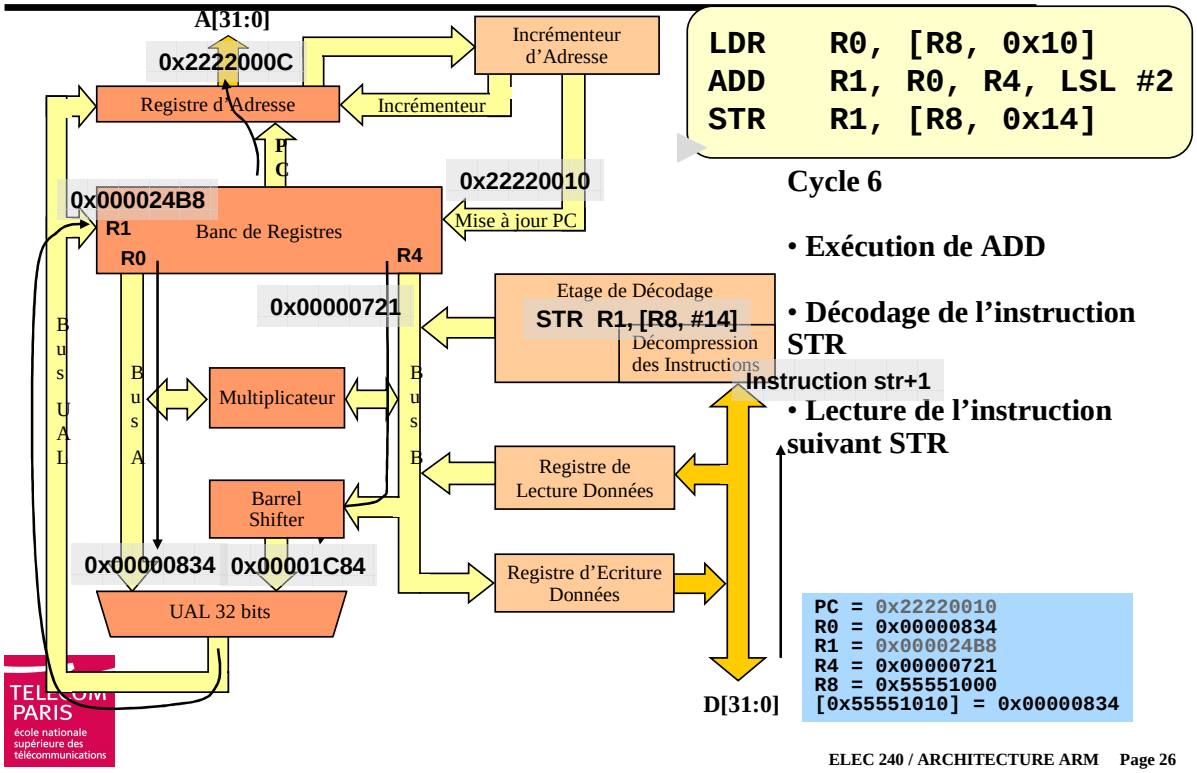
Séquence d'Instructions : Cycle 5



LDR n'est toujours pas terminé, il faut maintenant transférer la donnée lue du registre tampon vers le registre R0.

Les autres instructions sont toujours gelées.

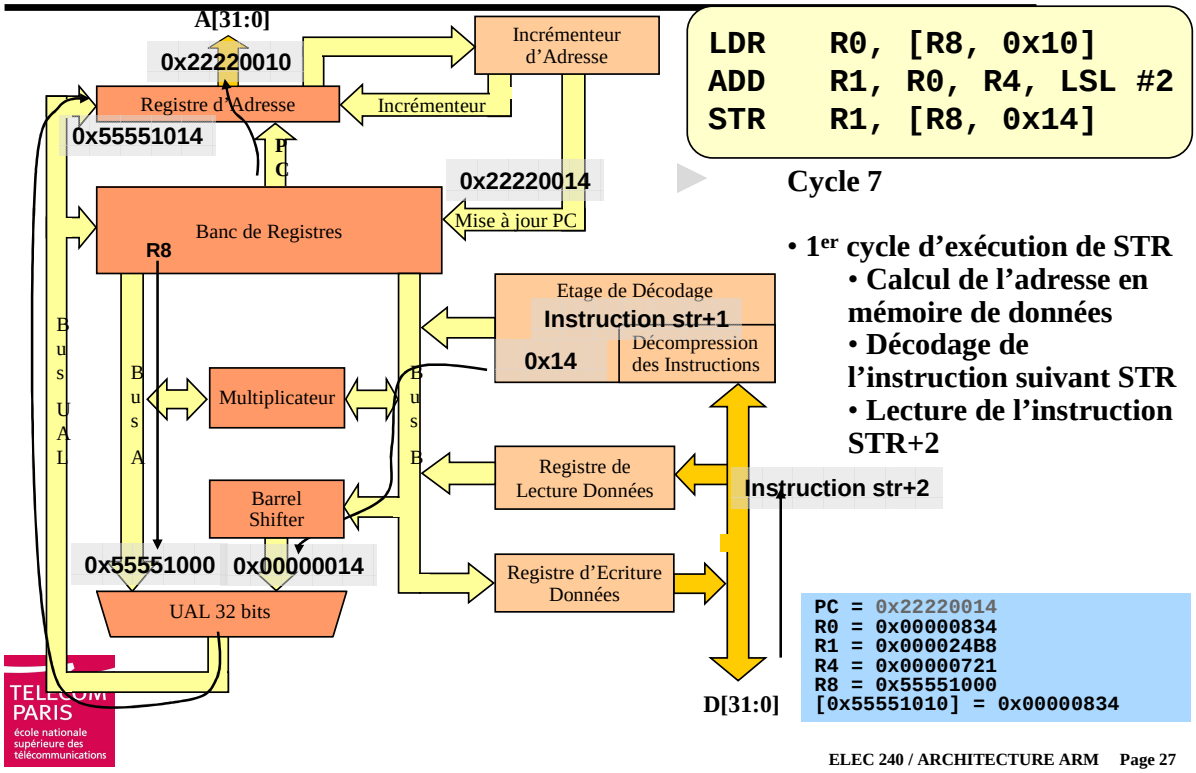
Séquence d'Instructions : Cycle 6



Le ADD est maintenant exécuté. Le registre R4 qui est sur le bus B est décalé de 2 bits par le barrelshifter

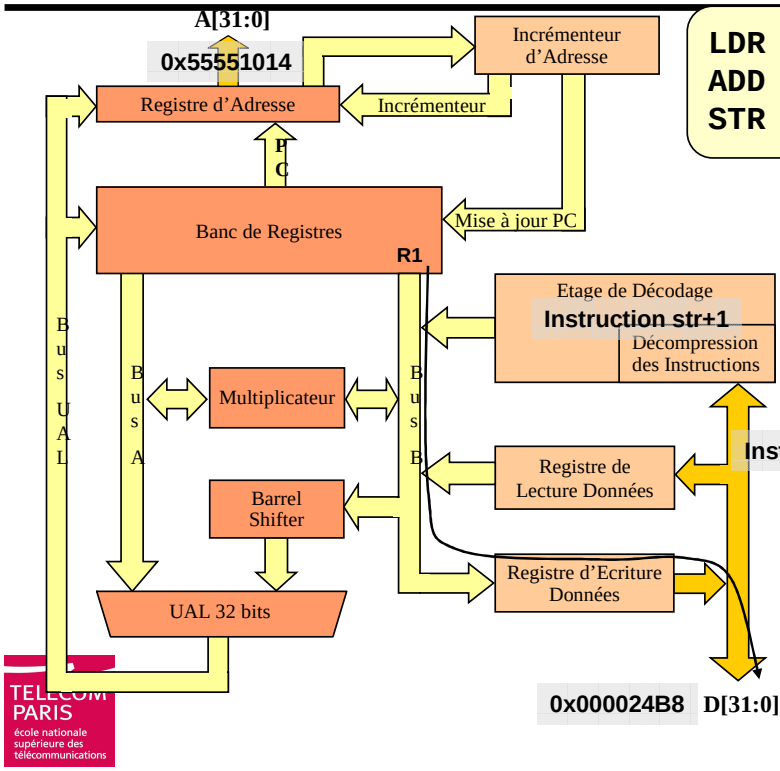
Le STR est décodé

Séquence d'Instructions : Cycle 7



Le STR est exécuté. Le calcul R8 + 0x14 est effectué dans le datapath

Séquence d'Instructions : Cycle 8




```
LDR   R0, [R8, 0x10]
ADD   R1, R0, R4, LSL #2
STR   R1, [R8, 0x14]
```

Cycle 8

- **2ème cycle d'exécution de STR**
  - **Ecriture en mémoire de données**

```
PC = 0x22220014
R0 = 0x00000834
R1 = 0x000024B8
R4 = 0x00000721
R8 = 0x55551000
[0x55551014] = 0x000024B8
```

TELECOM PARIS
école nationale supérieure des télécommunications

Le STR n'est pas terminé, le registre R1 est écrit dans le registre tampon avant d'être transférer à la mémoire (1 cycle de +)

Exceptions

L'activation d'une exception donne lieu au passage dans une mode particulier et au branchement dans un programme par le biais d'une table de vecteurs
7 sortes :

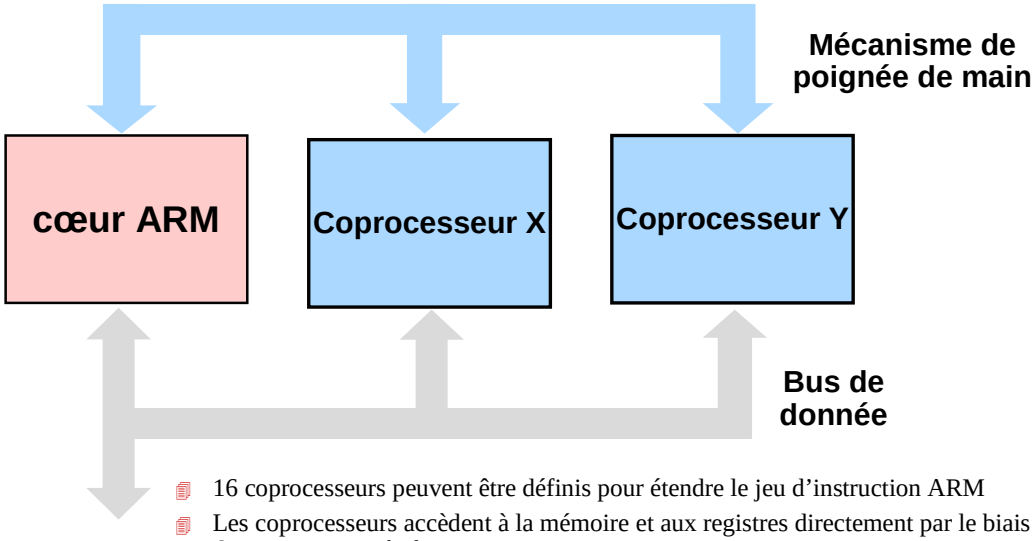
TYPE	MODE	VECTEUR	Retour en USER
RESET	Supervisor	0x00000000	Le CSPR et le PC sont restaurés en même temps
Instruction indéfinie	Undef	0x00000004	MOVS PC,r14
Interruption logicielle SWI	Supervisor	0x00000008	MOVS PC,r14
Problème Fetch instruction	Abort	0x0000000C	SUBS PC,r14,#4
Problème Fetch donnée	Abort	0x00000010	SUBS PC,r14,#8
Interruption matérielle IRQ	IRQ	0x00000018	SUBS PC,r14,#4
Interruption matérielle FIRQ	FIQ	0x0000001C	SUBS PC,r14,#4

TELECOM PARIS
école nationale supérieure des télécommunications

Pour passer du mode USER vers un mode avec privilèges, il faut nécessairement une exception qui va forcer le processeur à aller effectuer un saut dans la table des vecteurs et effectuer un programme de traitement spécifique.

Pour le retour en mode USER, il faut que les registres LR et SPSR soient simultanément restaurés dans les registres PC et CPSR. C'est le but de l'indicateur S de l'instruction de retour du sous-programme d'exception.

Coprocesseurs



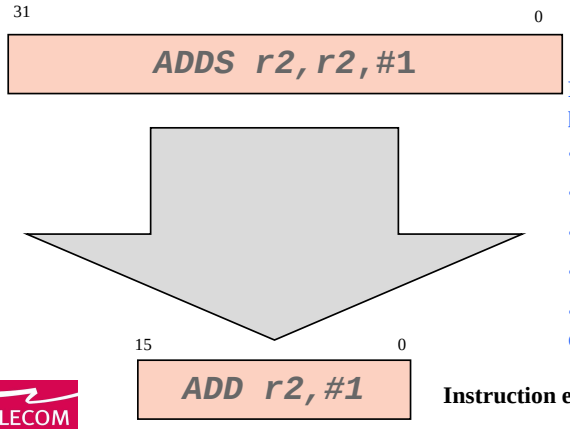
- 16 coprocesseurs peuvent être définis pour étendre le jeu d'instruction ARM
- Les coprocesseurs accèdent à la mémoire et aux registres directement par le biais d'instructions spécifiques

TELECOM PARIS école nationale supérieure des télécommunications

Il est possible d'adjoindre des accélérateurs de calcul ou "coprocesseur" au processeur ARM. Les coprocesseur sont pilotés par l'ARM via une liaison spécifique. Lorsqu'une instruction coprocesseur est décodée, l'ARM déclenche le coprocesseur adressé qui va lire ou écrire à la volée les données transitant sur le bus. Si aucun coprocesseur n'est détecté, le processeur rentre en mode "undefined instruction" qui lui permet d'effectuer le calcul sans coprocesseur donc plus lentement.

Mode Thumb

- **Le mode Thumb est un jeu d'instructions codé sur 16 bits**
  - ***Optimisé pour la densité de code à partir de code écrit en langage C***
  - ***Augmente les performances pour des espaces mémoires réduits***
  - ***Sous ensemble des fonctionnalités du jeu d'instructions ARM***
- **Le cœur a deux modes d'exécution des instructions: ARM et Thumb**
  - ***On passe d'un mode à l'autre en utilisant l'instruction*** BX



Instruction en mode ARM (32 bits)

Pour la plupart des instructions générées par le compilateur:

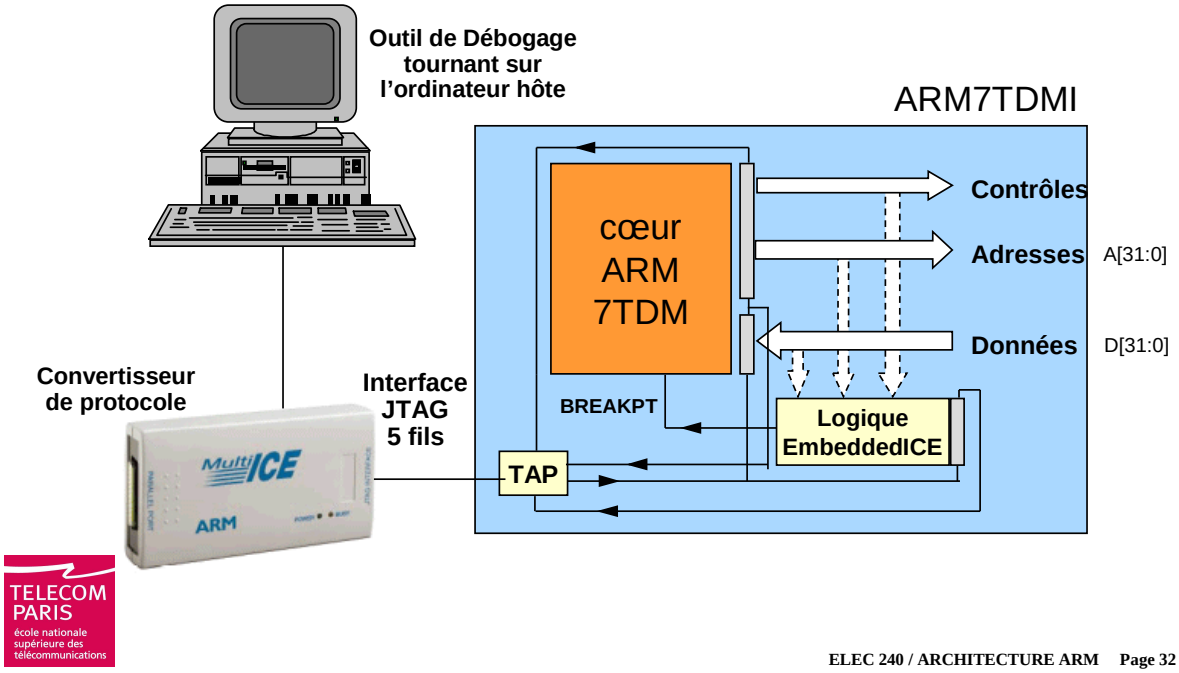
- **L'exécution conditionnelle n'est pas utilisée**
- **Les registres Source et Destination sont identiques**
- **Seul les premiers registres sont utilisés**
- **Les constantes sont de taille limitée**
- **Le registre à décalage n'est pas utilisé au sein d'une même instruction**

Instruction en mode Thumb (16 bits)

TELECOM PARIS
école nationale supérieure des télécommunications

La taille du code en mode thumb correspond typiquement à 65-70% de la taille du code en mode ARM.

Débogage Embarqué



TELECOM PARIS
école nationale supérieure des télécommunications

Il existe 3 chaînes de débogage sur le cœur ARM :

0 – E/S du cœur (incluant les données)

1 – données uniquement

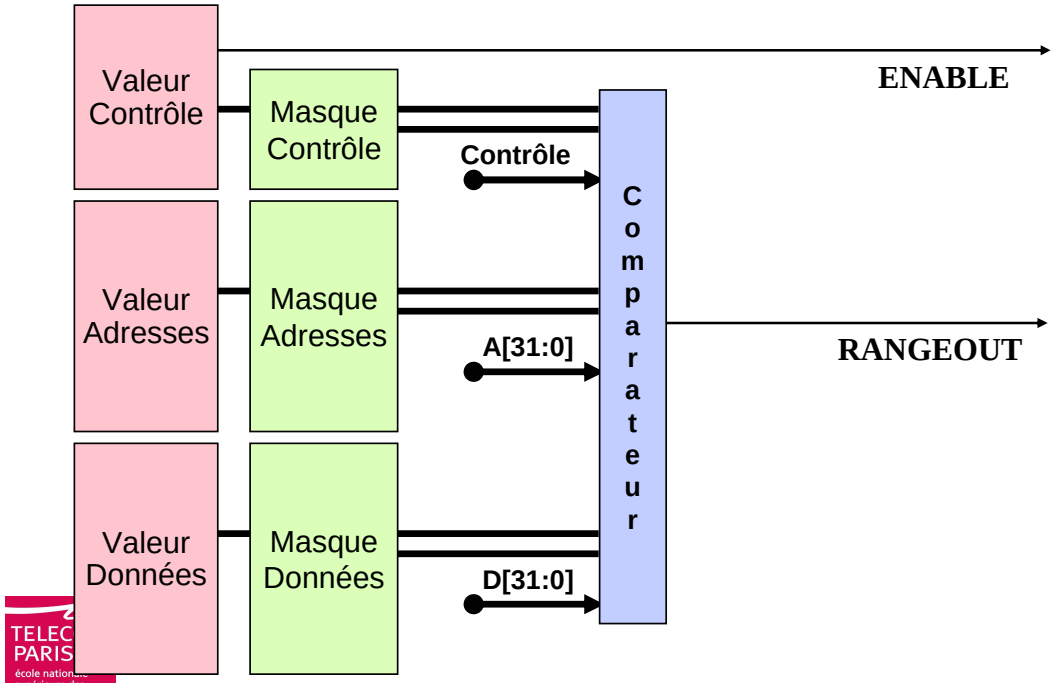
2 – emulateur embarqué

Le port JTAG permet d'accéder à ces 3 chaînes. Les données sont séparées pour aller plus vite dans la cas d'analyse des données seules.

L'émulateur embarqué "embedded ICE" analyse les évènements sur les bus et génère un point d'arrêt BREAKPT quand il y a comparaison avec un évènement préprogrammé. Dans ce cas le cœur est arrêté et islolé du reste du système. C'est au débogueur d'examiner et/ou changer les E/S par le biais des chaînes de débogage.

La communication avec l'émulateur embarqué s'effectue aussi par le JTAG en utilisant un convertisseur de protocole (MultiICE) .

Unité de Gestion des Points d'Arrêt



TELECOM PARIS école nationale supérieure des télécommunications

ELEC 240 / ARCHITECTURE ARM Page 33

Par le biais du JTAG, le concepteur peut accéder à la programmation de l'émulateur embarqué. Ceci consiste à définir la condition du point d'arrêt sur les bus de l'ARM.

L'émulateur consiste en une série de comparateur qui active le signal RANGEOUT qui va activé le point d'arrêt en quand la condition est remplie.

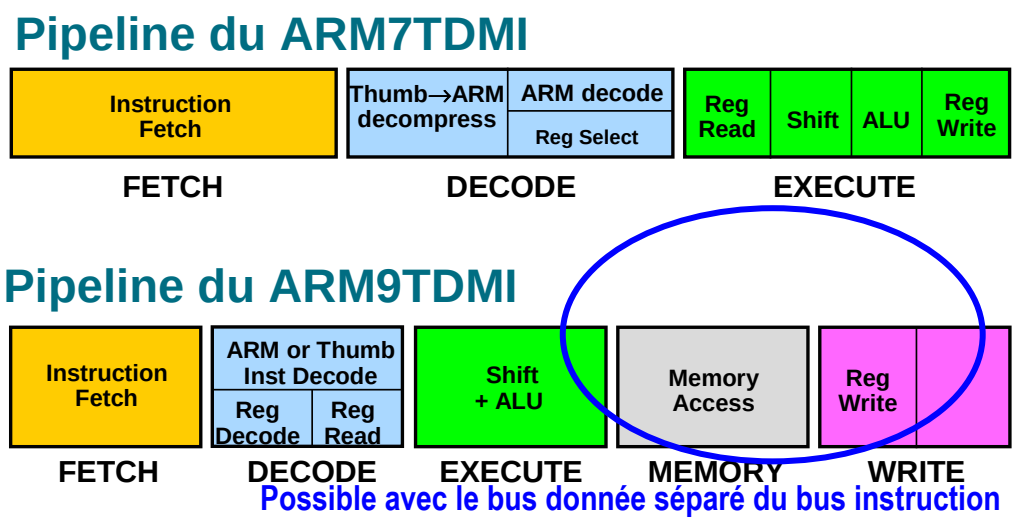
ARM9TDMI

- **Implémentation à double bus (architecture Harvard)**
  - ***Augmente la bande passante entre le microprocesseur et la mémoire***
    - **Interface mémoire Instructions**
    - **Interface mémoire Données**
  - ***Permet l'accès simultané aux mémoires Instructions et Données***
  - ***=> Modifications pour améliorer le nombre de cycles par instruction (CPI "Cycles Per Instruction") jusqu'à ~1.5***
- **5 niveaux de pipeline**
  - ***=> Modifications pour améliorer la fréquence maximum de l'horloge***

TELECOM PARIS
école nationale supérieure des télécommunications

La grosse différence consiste à passer d'une architecture Von Neuman à une architecture Harvard. Le fait de disposer des bus instructions/données séparés permet de modifier le pipeline pour avoir une phase d'accès à la mémoire donnée sans rupture de pipeline avec les instructions LOAD/STORE

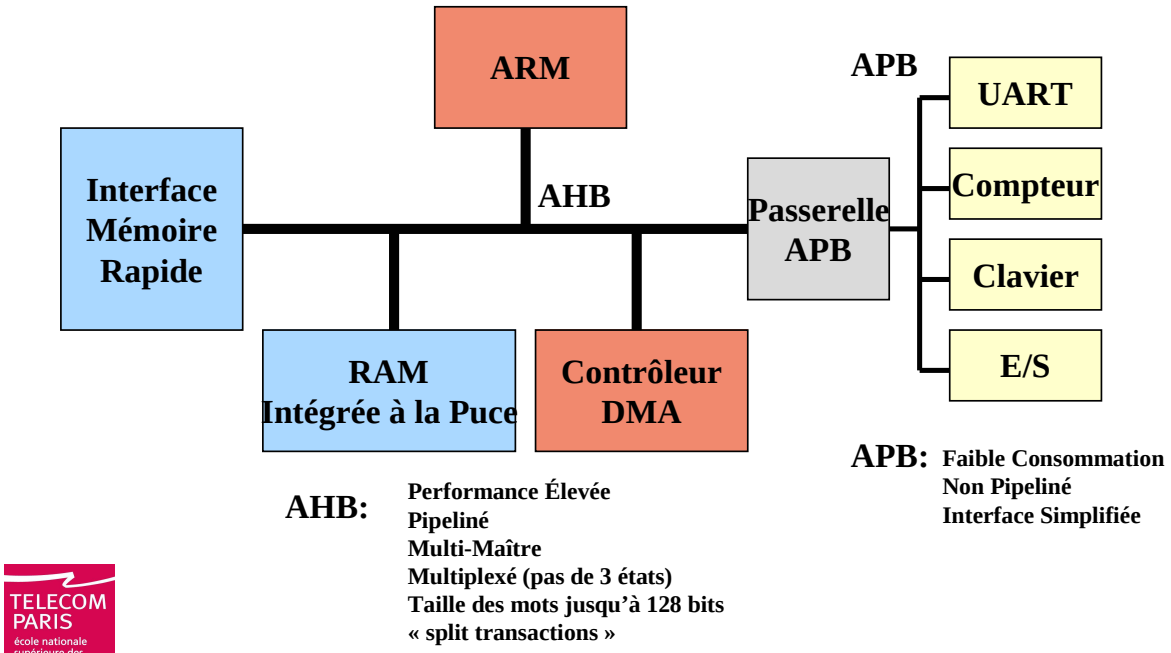
Modifications du Pipeline pour le ARM9TDMI



Dans l'ARM9, les 2 phases supplémentaire d'accès mémoire MEMORY et de transfert avec le registre tampon WRITE permettent d'effectuer des LOAD/STORE sans rompre le pipeline. Les 2 cycle perdus de l'ARM7 lors des LOAD/STORE ont maintenant disparus.

Comme l'ARM 9 dispose d'une architecture Harvard, le FETCH d'une nouvelle instruction peut se faire simultanément avec l'accès mémoire MEMORY.

le Bus AMBA



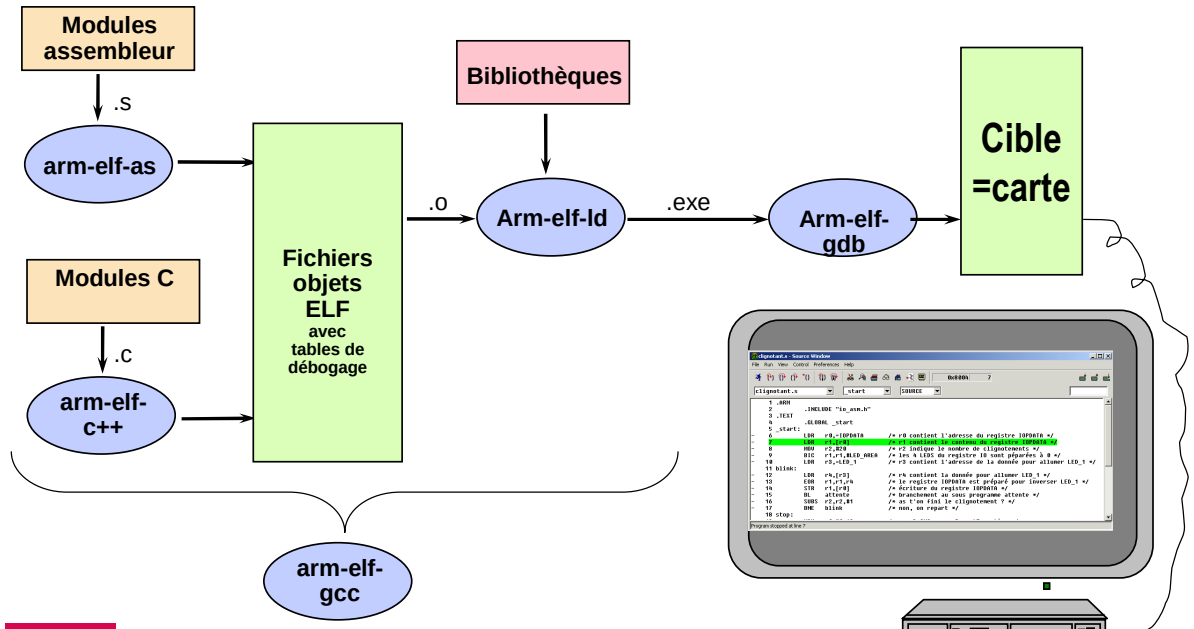
TELECOM PARIS école nationale supérieure des télécommunications

A côté du processeur, la société ARM propose le bus AMBA qui permet de concevoir une système complet. Ce bus AMBA est en fait composé de 2 bus :

- Un bus rapide Advanced High Bus
- Un bus lent Advanced Peripheral Bus.

Le bus AHB permet de connecter des périphériques rapides comme les contrôleurs SDRAM, DMA, vidéo,…. Le bus APB permet de s'interfacer avec des périphériques lents comme un clavier, des E/S ne dépassant pas 1Mbit/s

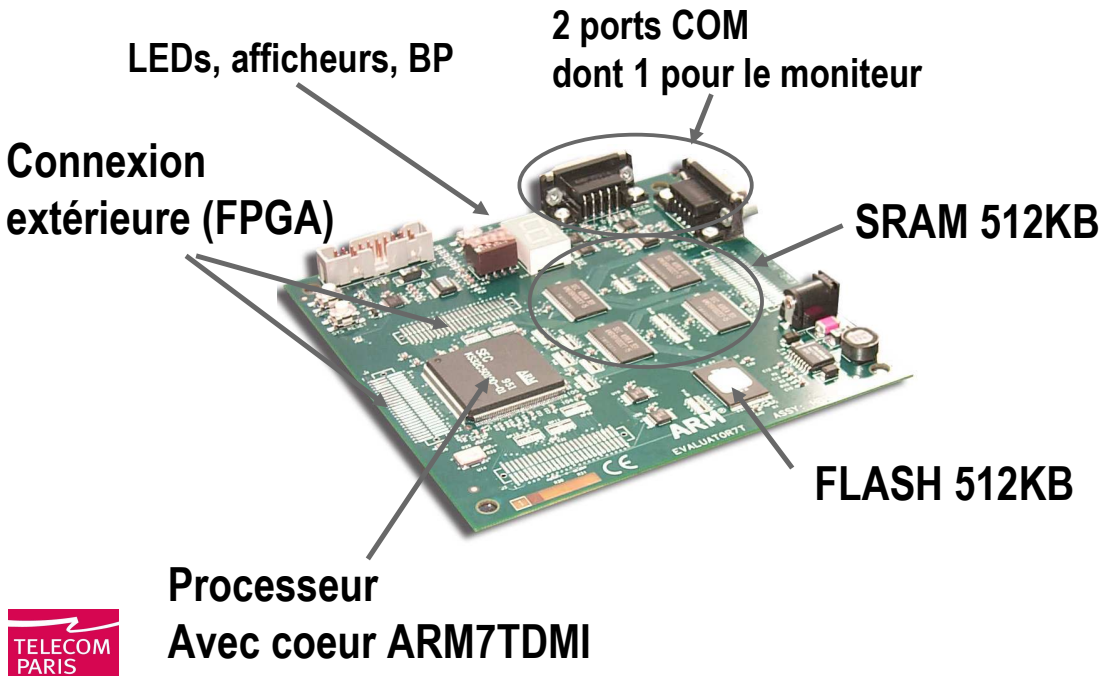
Développement logiciel



TELECOM PARIS
école nationale supérieure des télécommunications

Les outils libres GNU sont utilisés. Le résultat de compilation est téléchargé sur la carte cible qui contient un circuit avec un cœur de processeur ARM.

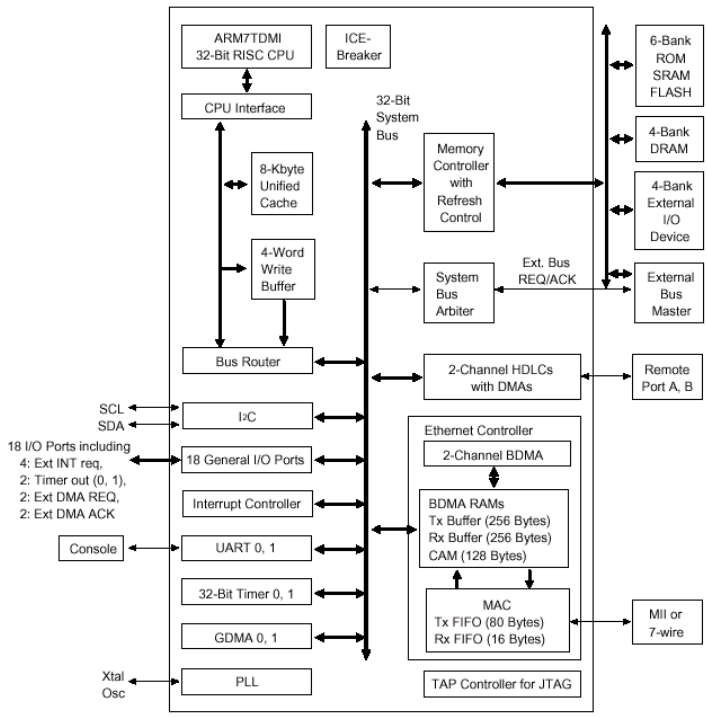
La Carte Evaluator 7T



TELECOM PARIS
école nationale supérieure des télécommunications

La carte Evaluator 7T contient un circuit de communication SAMSUNG KS32C50100 intégrant un cœur ARM. Cette carte dispose de 512K de mémoire SRAM et de mémoire FLASH préchargée avec le moniteur Angel et un moniteur de démarrage.

Circuit SAMSUNG KS32C50100

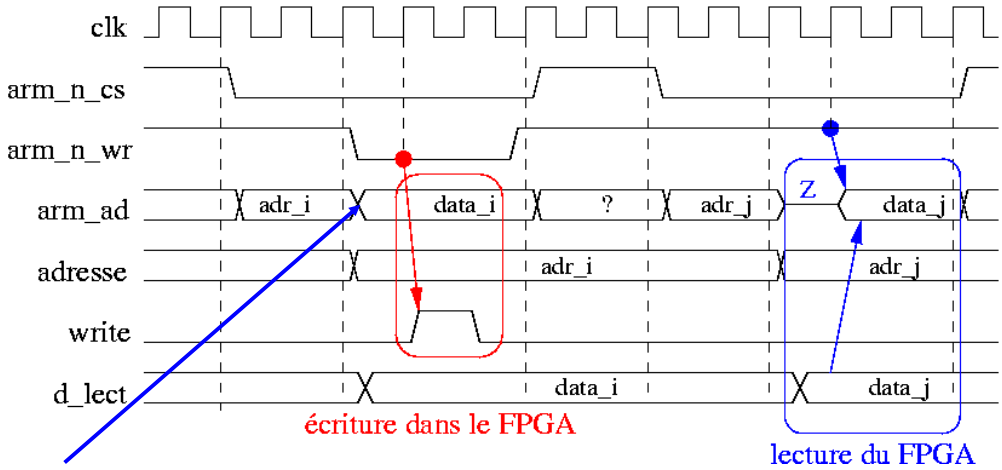


TELECOM PARIS
école nationale supérieure des télécommunications

Autour du cœur ARM, le processeur SAMSUNG contient de nombreux périphériques :

- 8K de mémoire rapide utilisable en local ou en cache
- 2 minuteries ou "timers"
- 2 liaisons série UART
- 2 canaux DMA
- 1 contrôleur mémoire supportant SRAM, DRAM et FLASH sur 6 bancs maximum
- 1 contrôleur ethernet
- 2 canaux HDLC

Exemple d'interface KS32C50100/FPGA



Bus adresse/données multiplexé

Adresse ARM = pointeur d'octets

0xA00014 = 1010 0000 0000 0000 0001 0100

N_CS — Adresse transmise au FPGA = pointeur de mots de 32 bits — Inutiles car toujours à 0

TELECOM PARIS école nationale supérieure des télécommunications

ELEC 240 / ARCHITECTURE ARM Page 40

Dans cet exemple le processeur utilise un protocole de communication synchrone nécessitant 5 cycles d'horloge.

La mappe mémoire est telle que le FPGA est dans le banc 5 du contrôleur mémoire entre les adresses 0xA00000 et 0xA10000. Le banc 5 est paramétré pour que les adresses et les données se partagent le bus. Attention les adresses sont des adresses de mot de 32 bits et non d'octets.

Lorsque le processeur s'adresse au FPGA , c'est-à-dire à l'adresse 0xA00000 par exemple, le signal arm_n_cs est actif à 0 durant 5 cycles. Le bus données arm_ad contient d'abord l'adresse puis les données. Le signal arm_n_wr est actif à 0 à partir du 3ème cycle dans le cas d'une écriture , sinon il reste à 1.

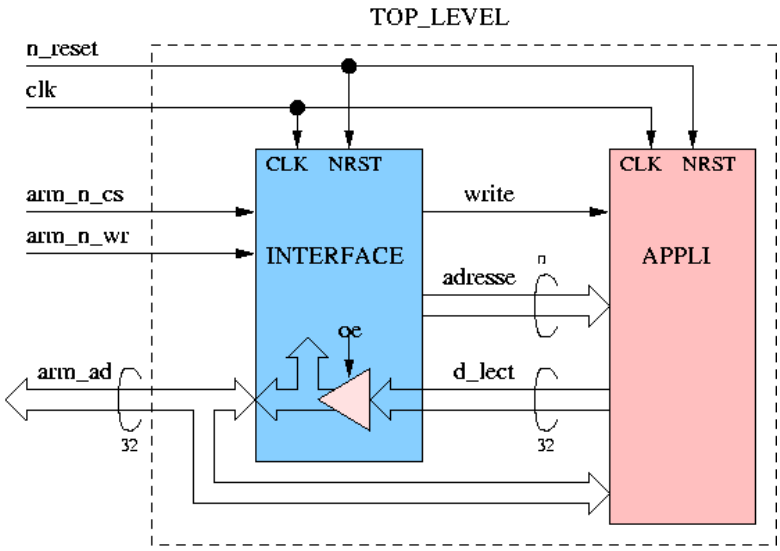
Le circuit d'interface doit effectuer les opérations suivantes :

• Extraire les adresses sur 15 bits

• Générer une impulsion write en cas d'écriture

• Faire passer la données lue d_lect sur le bus donnée durant les 2 derniers cycles dans le cas d'une lecture

Architecture de l'interface KS32C50100/FPGA



TELECOM PARIS
école nationale supérieure des télécommunications

L'architecture du FPGA peut être la suivante. Elle est décomposée en 2 blocs, un bloc d'interface générant et un bloc applicatif utilisant les signaux filtrés de l'interface comme décrit précédemment. Il faut noter la présence d'un buffer 3 états dans l'interface qui permet au processeur de lire les données d_lect. Le signal oe doit être actif durant les 2 derniers cycles.

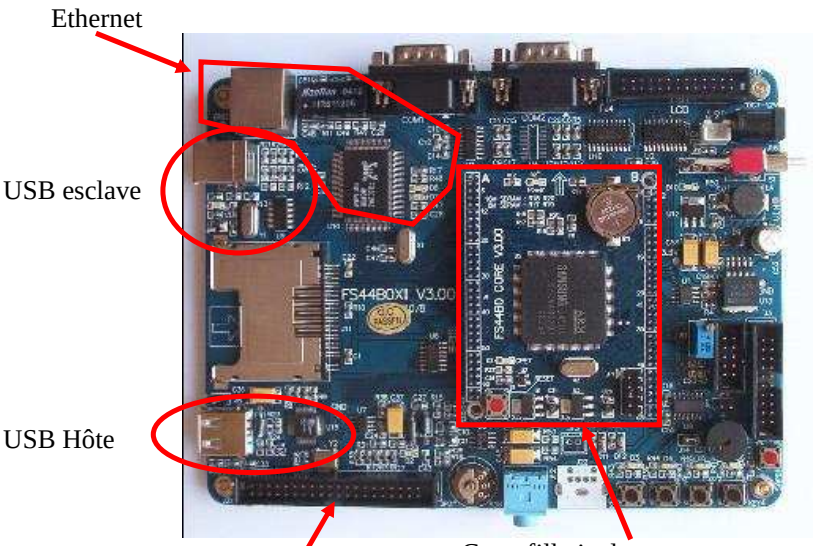
Plate-forme d'étude

Carte DE2 Carte FS44BOX2

TELECOM PARIS
école nationale supérieure des télécommunications

L'interface avec le FPGA va s'appuyer sur l'interface IDE.

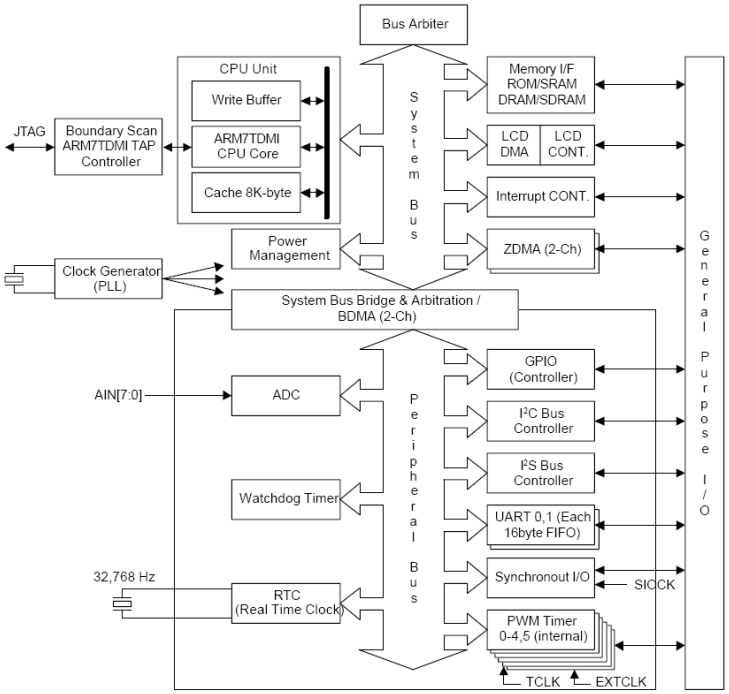
La Carte FS44 BOX II



TELECOM PARIS
école nationale supérieure des télécommunications

La carte dispose d'un processeur SAMSUNG S3C44BOX. Le système est articulé autour d'un noyau uClinux.

Circuit SAMSUNG S3C44BOX



TELECOM PARIS
école nationale supérieure des télécommunications

Le circuits SAMSUNG S3C44BOX dispose de nombreuses interfaces utiles à un système embarqué. C'est sur ce processeur que les TDs vont s'appuyer.