PROCESIM : étude d'un processeur simple en PC/PO

Groupe Architectures Logicielles et Matérielles

Document Etudiants

1 Définition du jeu d'instructions

1.1 Registres et format des instructions

Le jeu d'instructions du processeur définit 16 registres généraux (R0 à r15) et trois registres spéciaux : cpsr, spsr et slr. Le registre R4 est câblé à 0. Lorsque R4 est utilisé comme registre destination, le résultat du calcul ou la donnée lue en mémoire n'est pas mémorisé : seuls les indicateurs ZNCV sont éventuellement mémorisés.

La partie opérative du processeur comprend également deux registres qui ne font pas partie du jeu d'instructions : Rinstr et Tmp destinés au stockage de l'instruction courante et à l'adresse utilisée dans les instructions load et store.

Nom	Fonction particulière
R0 ou PC	Compteur ordinal
R1 ou SP	Pointeur/sommet de pile
R3 ou LR	Adresse retour de procédure
R4 ou 0	Constante zéro
R2,R5 à R15	néant

Nom	Fonction
cpsr	registre d'état courant
spsr	sauvegarde de cpsr (départ en it)
slr	adresse de retour (traitant d'it)

Les instructions sont toutes codées sur 16 bits (elles doivent être stockées en mémoire à des adresses paires et PC est incrémenté de deux après chaque instruction). Elles sont regroupées en quatre catégories, chacune étant doté d'un type de format binaire. Tout bit noté x est ignoré et doit être codé à 0.

Numéros des bits							
15-14	13	12	11-8	7-4	3-0	Instructions	
0 0	S	i	op	sou	dest	calculs	
0 1	E	i	val	depl	base	mémoire	(load/store)
1 0	Lbar	x	cond	depl		branchements	bcond/blcond
1 1	0	0	1101	spec	gen	spéciales	mrs : Rspec → Rgen
1 1	0	1	1101	gen	spec		msr : Rspec ← Rgen
1 1	1	1	xxxx	xxxx	xxxx		rti
1 1	1	0	0xxx	xxxx	xxxx		ignore
1 1	1	0	1xxx	xxxx	xxxx		accept
à définir							swi

Les instructions mrs et msr permettent les transferts entre registres spéciaux et généraux. Les autres instructions spéciales sont liées au mécanisme d'interruption.

1.2 Instructions de calcul

Le champ op indique la nature du calcul réalisé. A noter : certaines opérations ne sont pas gérées dans l'environnement de travaux pratiques avec PROCESIM.

Nom	op	ok TP	Opération
SUB	0000	oui	sub
ADC	0001	non	add with Carry
RSB	0010	non	reverse sub
RSC	0011	non	rev sub with Carry
SBC	0100	non	sub with Carry
MVN	0101	oui	move not
BIC	0110	oui	bit clear (andnot)
XNOR	0111	oui	xor not

Nom	op	ok TP	Opération
LSL	1000	non	shift «0
LSR	1001	non	shift 0»
ASR	1010	non	shitf signe»
OR	1011	non	or
ADD	1100	oui	add
MOV	1101	oui	mov
AND	1110	oui	and
XOR	1111	oui	xor

L'opérateur de décalage de l'opérande droit n'est pas disponible. Il est remplacé par trois instruction spécifiques de décalage. Il n'est pas possible de spécifier séparément l'opérande gauche et la destination du résultat d'un calcul : tous deux sont stockés dans le même registre et les calculs sont de la forme $Rdest \leftarrow Rdest\ op\ Droit$.

Les instructions $sub\ r6, r7$ et $lsr\ r3, \#4$ de ce processeur correspondent aux instructions ARM $sub\ r6, r6, r7$ et $mov\ r3, r3, LSR\#4$.

Selon le bit i, l'opérande droit est soit une constante entière #sou immédiate (codée sur quatre bits dans l'instruction) pour i=1, soit le contenu du registre *Rsou* pour i=0. Le bit S indique si les indicateurs NZCV de cpsr doivent être mis à jour.

1.3 Accès à la mémoire : Load et Store

L'adresse mémoire utilisée (spécifiée entre crochets) est la somme du contenu du registre *Rbase* et d'un déplacement. Ce déplacement est analogue à l'opérande droit d'une instruction de calcul : #depl ou *Rdepl* selon le bit i.

Les instructions ldr et str ne gérent qu'une seule taille de transfert (pour les travaux pratiques avec PROCESIM, il s'agira de l'octet).

Le bit E (écriture) indique s'il s'agit d'une instruction ldr ($E = 0 : Rval \leftarrow Mem[Rbase + depl]$) ou str ($E = 1 : Rval \rightarrow Mem[Rbase + depl]$).

1.4 Branchements Bcond et BLcond

Par défaut, pour l'étude en travaux dirigés, nous supposerons que les conditions testables et leur codage sont identiques à celles offertes par un processeur ARM. Pour les travaux pratiques, seules trois conditions seront gérées, avec une interprétation différente du champ cond.

Le déplacement relatif au compteur ordinal, exprimé en octets, est codé sur un octet et considère que le pc a une avance d'une instruction. $Lbar = 0$ indique un branchement avec sauvegarde de pc dans le registre LR (R3). $Lbar = 1$ indique un saut ordinaire. **Attention : c'est l'inverse du codage binaire en ARM.**

	Cond	
Nom	TD	TP
EQ	0000	10xx
NE	0001	—-
CS	0010	01xx
CC	0011	—-

	Cond	
Nom	TD	TP
MI	0100	—-
PL	0101	—-
VS	0110	—-
VC	0111	—-

	Cond	
Nom	TD	TP
HI	1000	—-
LS	1001	—-
GE	1010	—-
LT	1011	—-

	Cond	
Nom	TD	TP
GT	1100	—-
LE	1101	—-
AL	1110	0000
NV	1111	—-

1.5 Interruptions

Il n'y a qu'un seul mode d'exécution et les registres lr et sp ne sont pas dédoublés. Lors d'un départ en interruption, le compteur ordinal pc et le registre d'état courant cpsr sont (les seuls registres) sauvegardés automatiquement (dans les registres slr et spsr). De plus, le bit I de cpsr (à ne pas confonfre avec le bit I dans le codage des instructions) est mis à 1, interdisant la prise en compte du signal extern IRQ.

Ensuite, le compteur ordinal est initialisé avec une constante (adresse) correspondant au vecteur d'interruption. Ce vecteur devra contenir une instruction de branchement vers le traitant.

L'instruction rti de retour d'interruption effectue le branchement de retour en fin de traitant et restaure cpsr à sa valeur d'avant l'interruption.

L'instruction SWI (software interrupt) est une instruction qui permet au programmeur de forcer explicitement un départ en interruption (même si le bit I est à 1). L'unique différence entre un départ en interruption déclenché par le signal IRQ avec I=0 et un départ déclenché par une exécution de l'instruction SWI est le vecteur d'interruption utilisé (Reset : 0, IRQ : 2, SWI : 4).

On peut forcer à 0 ou à 1 le bit I de cpsr avec une séquence mrs ; bic ou or, msr. Les instructions accept (ou encore clri, $I \leftarrow 0$) et ignore (ou encore seti, $I \leftarrow 1$) permettent de le faire en une seule instruction (opération atomique).

2 Réalisation en circuit PC/PO

2.1 Organisation générale

L'organisation générale du processeur est décrite figure 1. La partie commande est un automate de Mealy si l'on utilise la paramétrisation, et de Moore sinon. Le prochain état dépend :

1. des bits du registre d'instruction spécifiant la nature de l'instruction.
2. pour les branchements conditionnels, des indicateurs NVCZ de cpsr et des bits (cond) du registre d'instruction. On peut utiliser un circuit combinatoire de paramétrisation générant directement un booléen indiquant si la condition de saut est vérifiée ou non.
3. du bit I de cpsr et du signal IRQ pour la gestion des interruptions.

Les sorties sont les commandes vers la mémoire (lire et écrire) et vers la PO, ces dernières étant générées directement (exemple : commande du multiplexeur sur le bus C) ou via un circuit combinatoire générant directement les signaux de commandes paramétrés par les champs op, sou ou dest du registre d'instruction (exemple : commandes de chargement des registres Rx ($x \in [0, 15]$) en fopnction du champ dest de l'instruction.

Il s'agit de dessiner le graphe de la PC. Dans chaque état, nous noterons une microaction, qui devra être traduite en valeurs des signaux de commandes vers la PO et la mémoire.

2.2 Commandes vers la partie opérative et micro-actions possibles

Nous ne détaillons pas ici TOUTES les commandes. Le principe est qu'il y a :

1. sortie d'un registre $R_i, i \in [0, 15]$ ou $R_{temporaire}$ vers bus B, ou bus A
2. sortie de valeur immédiate (partie de Registre Instruction) ou de cpsr, vers bus B
3. choix de l'opération dans l'UAL
4. chargement d'un registre (ne pas oublier le Registre Instruction)
5. mise en communication des deux bus B et bus données
6. le multiplexeur du bus C : sortie UAL ou bus de données
7. le multiplexeur du bus d'adresses : sortie UAL ou registre PC.
8. concernant le registre d'état, il est possible de le charger en entier depuis le bus C, ou charger seulement NZCV depuis l'UAL ou forcer à 0 ou à 1 le bit I.

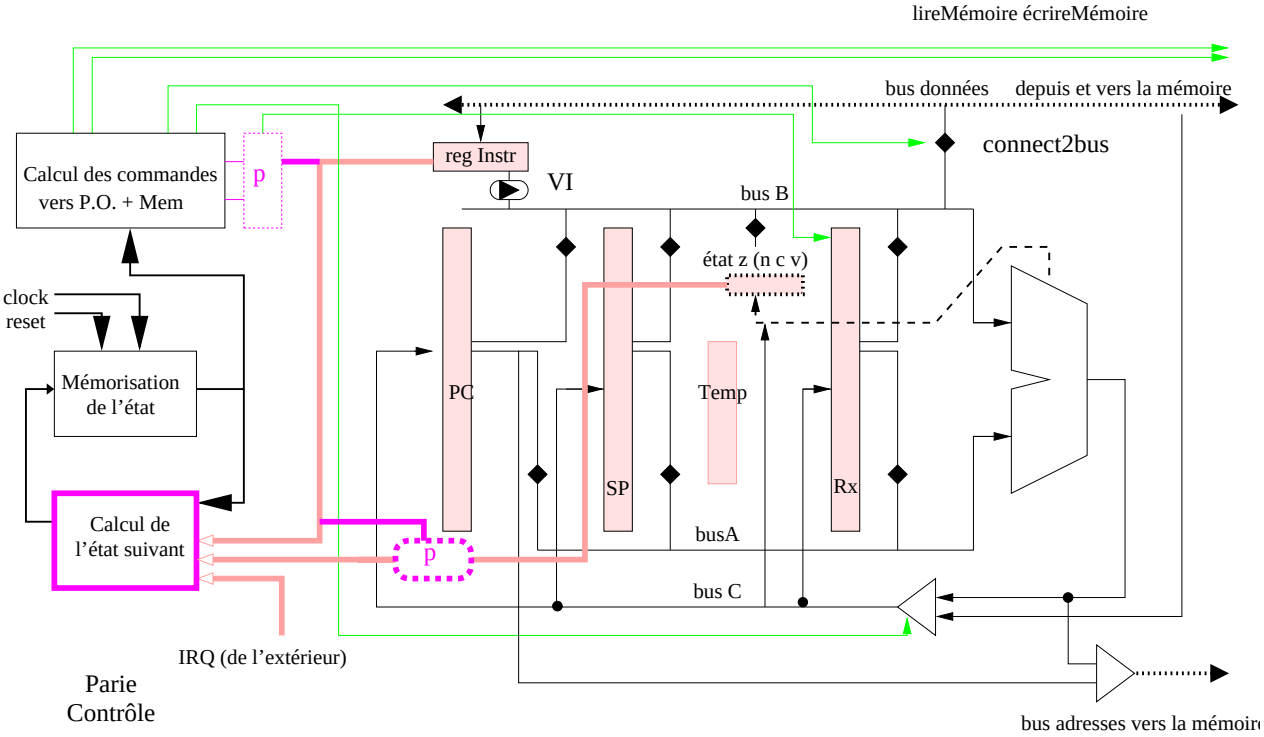

FIGURE 1 – Vue générale du processeur (p : paramétrisation)

Voici quelques exemples de microactions :

- RInst $\lll$ MEM[PC] : sortir PC sur bus adresses, lire mémoire, charger Rinstr (une copie de l'instruction lue transite sur le bus de données).
- PC ++ : sortir PC sur un bus (A ou B), opération ual : +2 sur entrée A ou B (une instruction occupe deux (adresses d') octets).