Architecture MIM1 2001/2002
Jean Duprat, Florent de Dinechin

TD $n+1$ à $n+2$ ou 3

Un microprocesseur RISC 16 bits

(corrigé sommaire)

1 Le jeu d'instructions

Rappel des épisodes précédents : on a choisi une machine à une adresse et accumulateur, à 32 registres (notons les R0 à R31). Le PC est, mettons, R31. Mon idée géniale du moment est que l'accumulateur soit R0. Les opérations arithmétiques et logiques rangent leur résultat dans l'accumulateur, et prennent comme argument l'accumulateur et soit un registre, soit une constante codée sur 5 bits, non signée, et complétée par des 0. Les opérations d'accès mémoire transferent le contenu de l'accumulateur de/vers la mémoire, à l'adresse donnée par un registre ou une constante de 5 bits.

Et voici le format du mot d'instruction.

- bits 0 à 4 : numéro du registre, ou constante, suivant le bit 5.
- bit 5 : dit si le contenu des bits 0 à 4 sera considéré comme une constante ou comme un numéro de registre.
- bits 6 à 9 : codage de la condition sous laquelle l'instruction sera exécutée.
- bits 10 : dit si les drapeaux doivent etre mis à jour par cette instruction.
- bits 11 à 15 : codage de l'instruction (32 instructions possibles).

Les instructions retenues sont les suivantes :

Instruction	Code	Action/commentaire
AND	0	
ORR	1	
XOR	2	
NOT	3	Acc ← not(Rx/x)
ADD	4	
SUB	5	Acc ← Acc - Rx/x
ADC	6	Acc ← Acc + Rx/x + Carry
SBC	7	
SBR	8	Acc ← Rx/x - Acc
	9	
	10	
	11	
LSL	12	
LSR	13	
LSR	14	
ASR	15	

Instruction	Code	Action/commentaire
LDA	16	Acc ← [Rx/x]
STA	17	[Reg/Cst] ← Acc
MTA	18	Acc ← Rx/x
MAR	19	Rx ← Acc ; constante interdite
JRP	20	PC ← PC+Rx/x
JRN	21	PC ← PC-Rx/x
JPA	22	
JSR	23	
RET	24	
	25	
	...	
	27	
	28	
	29	
	30	
	31	

Les conditions codées sur 4 bits sont AL (always), O (overflow), NO (no overflow), C, NC, Z, NZ, P, NP, PN (=P ou Z), N (= non P et non z), ...

Un mnémonique est de la forme

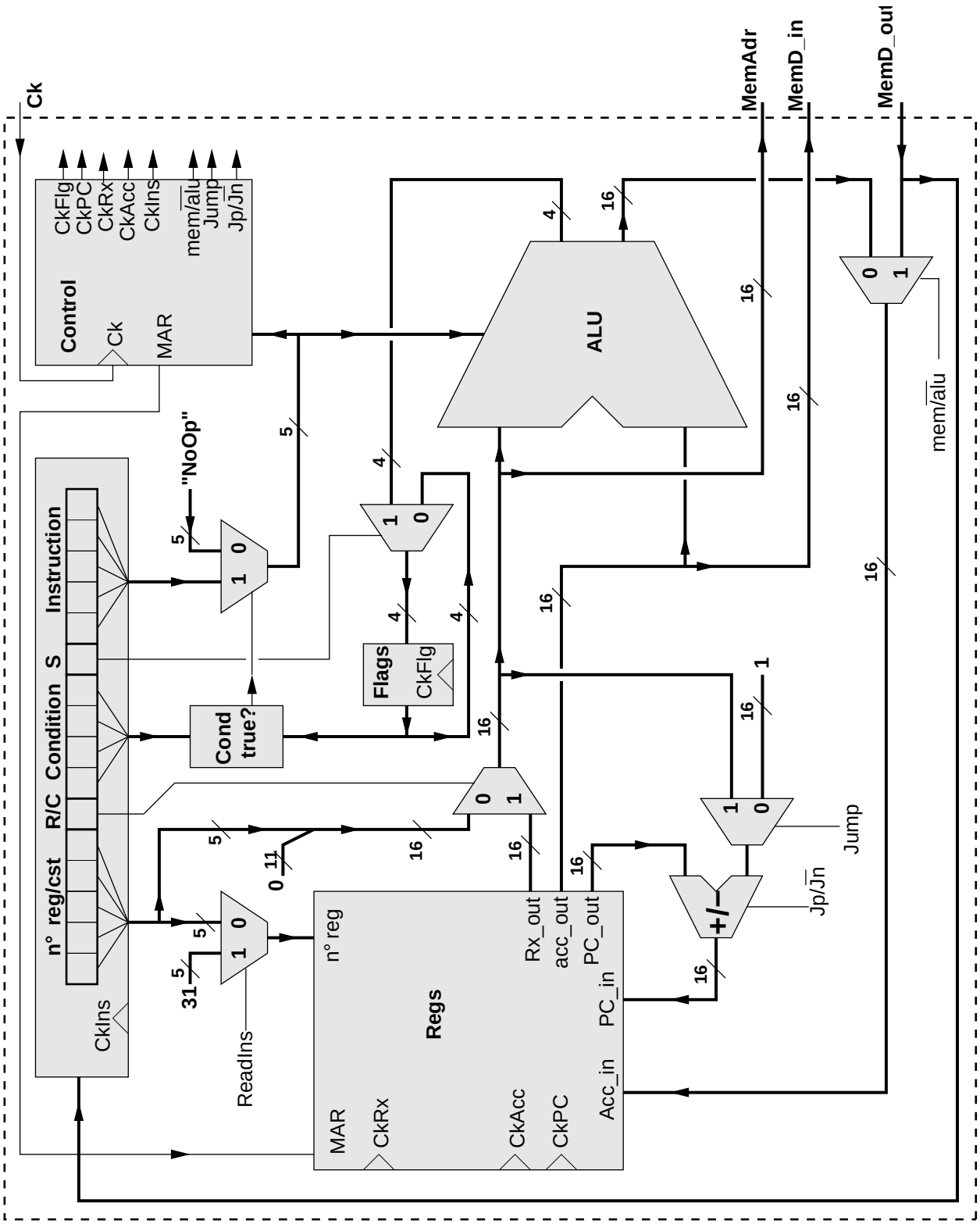
```
INS [condition] [S] Rx
```

ou bien

```
INS [condition] [S] #x
```

dans lequel `INS` est une instruction, `condition` est une condition optionnelle (par défaut AL), `S` signifiant qu'il faut mettre à jour les drapeaux, `Rx` donnant le registre et `#x` la constante.

2 Dessin global



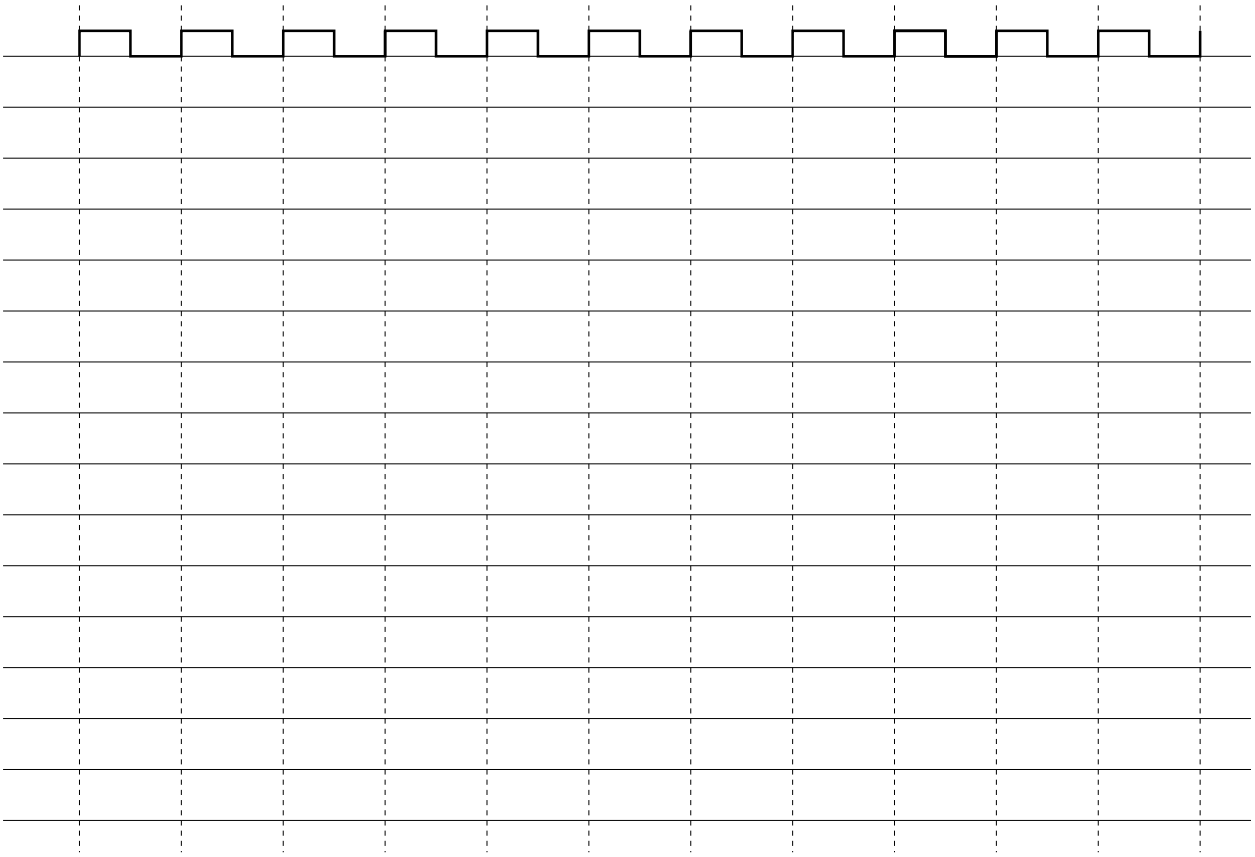
Le banc de registres sort toujours Acc et un autre registre Rx. Toutes les instructions écrivent dans Acc sauf l'instruction MAR, qui nécessite un traitement à part car elle écrit dans Rx. C'est la raison des deux signaux MAR et CkRx. La complexité additionnelle est lâchement cachée dans le banc de registres.

On a supposé que mémoire et banc de registres fonctionnent comme suit (soyez conscient que ce fonctionnement a peu à voir, pour des tas de raisons technologiques, avec ce qui se vend chez Darty. Mais on sait fabriquer ce type de mémoire au moyen des TPs précédents) :

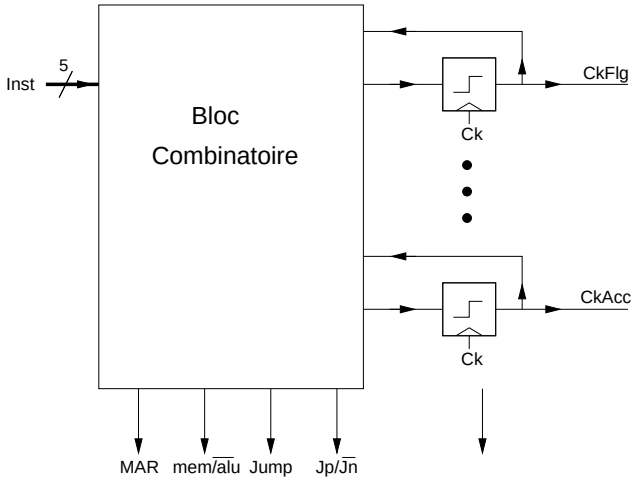
- En lecture, ils se comportent comme un bloc combinatoire, c'est-à-dire que les données en sortie arrivent après que l'on a présenté l'adresse en entrée, sans qu'il soit besoin de signal d'horloge supplémentaire.
- En écriture, les données présentes sont stockées à l'adresse présentée en entrée au front montant du signal d'horloge.

Exécution conditionnelle : le bloc de commande, en recevant le code instruction "NoOp", ne provoque aucun front montant sur les différentes horloges rangeant des résultats dans les registres, la mémoire ou les drapeaux. Dans ce cas, il met aussi Jump à 0 et JB/Jn à 1 pour que le PC soit incrémenté normalement.

3 Chronogrammes



4 L'automate



5 En bonus, le processeur de l'an dernier (à 2 adresses)

5.1 Le jeu d'instructions

On a choisi une machine à deux adresses, à 16 registres R0 à R15, plus un PC. Les opérations arithmétiques et logiques opèrent soit sur deux registres, soit sur un registre et une constante codée sur 4 bits, non signée, et complétée par des 0. Les opérations d'accès mémoire sont à deux registres : adresse et donnée. L'adresse peut être une constante de 4 bits.

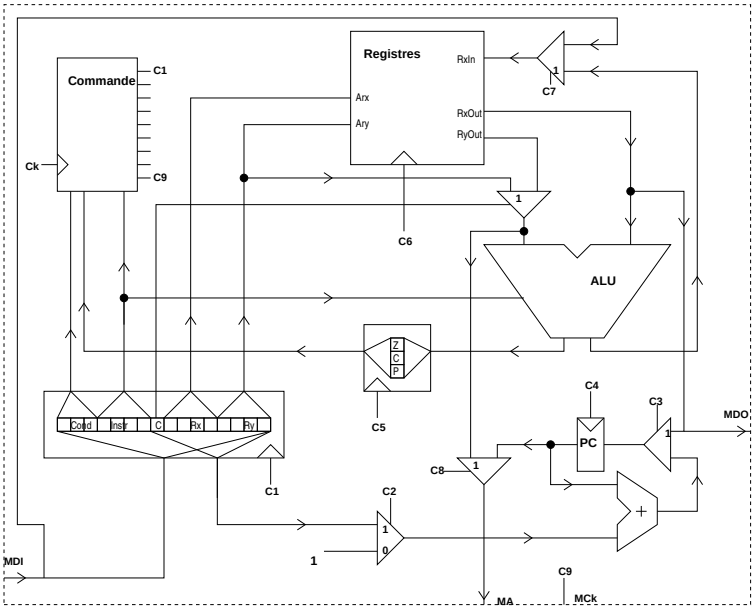
Et voici le format du mot d'instruction tel que l'on peut le voir sur la figure au verso.

- bits 0 à 2 : codage de la condition sous laquelle l'instruction sera exécutée.
- bits 3 à 6 : codage de l'instruction (16 instructions possibles).
- bit 7 : détermine si le second opérande est une constante ou un registre.
- bits 8 à 11 : numéro du premier registre Rx. Pour les opérations arithmétiques ou logiques, le résultat est stocké dans le premier registre après exécution. Pour les opérations mémoire, l'adresse est le second registre, la donnée est le premier registre.
- bits 12 à 15 : numéro du second registre Ry, ou constante, suivant le bit 7.

Les instructions retenues sont les suivantes :

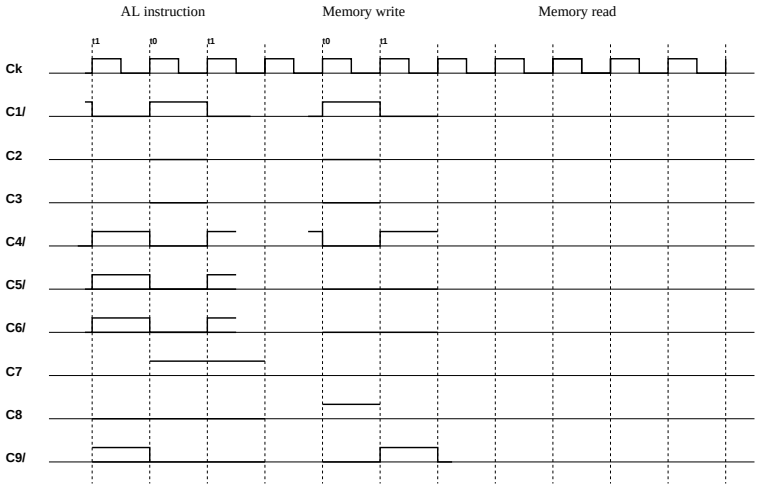
Instruction	Code	Commentaire
`ADD Rx, Ry/C`	0	
`SUB Rx, Ry/C`	1	
`CMP Rx, Ry/C`	2	
`MOV Rx, Ry/C`	3	
`AND Rx, Ry/C`	4	
`OR Rx, Ry/C`	5	
`XOR Rx, Ry/C`	6	
`NOT Rx, Ry/C`	7	Rx←not(Ry) / not C
`LSL Rx, Ry/C`	8	
`LSR Rx, Ry/C`	9	
`ASR Rx, Ry/C`	10	
	11	"reserved for future extensions"
`JR dep`	12	déplacement signé codé sur les bits 7 à 15
`JA Rx`	13	
`LDR Rx, [Ry/C]`	14	
`STR Rx, [Ry/C]`	15	

5.2 Dessin global et conventions



Le banc de registres est à deux ports RxOut et RyOut en lecture et un port en écriture (RxIn).

5.3 Chronogrammes



Les parties laissées blanches signifient que la valeur du signal est sans importance. Je vous laisse le soin de compléter ces chronogrammes.

5.3.1 Le cycle de Von Neumann

Entre t1 et t0, le PC est recopié (C8=0) sur le fil MA (bus d'adresse de la mémoire), la mémoire extrait le mot d'instruction à cette adresse et le présente sur le bus MDI. à l'instant t0 ce mot d'instruction est stocké dans le registre d'instruction (front montant sur C1/) ce qui permet à l'instruction de s'exécuter entre t0 et t1.

Le PC est soit incrémenté, soit modifié par un saut. Pour accomoder ces deux cas il faut que le front montant de C4/ soit à la fin de l'exécution de l'instruction, donc à t1.

Pour toutes les instructions on a donc C1/ et C4/ en opposition de phase.

5.3.2 Remarques sur le décodage des instructions

Il semble que C2, C3, C7 dépendent uniquement des quatre bits du mot d'instruction codant l'instruction. Si l'on en fait une telle fonction combinatoire, leur valeur sera fixée jusqu'au t0 suivant. Est-ce un problème ?

5.3.3 Instructions arithmétiques et logiques

On stocke en même temps le résultat de l'opération et les drapeaux, donc C5/ etC6/ sont toujours identiques. La présence d'un front montant dépend du décodage de la condition entre t0 et t1.

5.3.4 Lecture mémoire

Il s'agit juste d'envoyer sur le bus d'adresse l'adresse à lire (C8=1) provenant soit des registres soit de la constante immédiate. À t1 on stocke la valeur provenant de la mémoire (c7=0) dans le banc de registre par un front montant sur C6/. On ne modifie pas les drapeaux (C5 reste à 0).

5.3.5 Écriture mémoire

Comme précédemment l'adresse est positionnée par C8=1. La valeur du registre à écrire en mémoire est extraite combinatoirement du banc de registre et placée sur le bus de données MDO. Il ne reste qu'à placer un front montant sur C9.

5.3.6 Sauts

Un saut ne modifie pas les registres ni les drapeaux ni la mémoire, donc C5, C6 et C9 restent à 0. C2 et C3 sont positionnés suivant que le saut est absolu ou relatif.