ESIR
Université de Rennes 1

15 janvier 2016

ESIR1 - ARC1 - CC2

Durée : 2 heures
Tous documents autorisés **sauf des documents liés aux TP et projet (y compris le dernier cours du 4 janvier)**.

Problème 1 (20p).

On souhaite construire un petit processeur défini par un jeu d'instructions présenté sur la figure 1. Le programme et les données sont stockés dans la mémoire «*RAM*» composée des 256 cellules de 16b. Dans ce processeur les opérations arithmétiques sont effectuées sur les opérandes venant du banc des registres « *Register Files* » et les résultats sont mémorisés dans le même banc des registres. Le banc des registres est composé des 16 registres de 16 bits. Les deux registres sont disponibles en lecture et un registre est accessible en écriture en parallèle. Le processeur communique avec les périphériques par un bus « *Data_Bus* » composé de : 16 bits des données, 8 bits d'adresses « *Address_Bus* », un signal « *IORD* » (pour signaler une opération « in ») et un signal « *IOWR* » (pour signaler une opération « out »). Pour construire ce processeur, utiliser entre autres : la mémoire « *RAM* », banc des registres « *RF* » « *Register File* », décodeur de l'instruction « *Decoder* », registre « *Register* », comparateur « *Comparator* » et l'additionneur « *Add* ». La composition d'une instruction est présentée sur la figure 2.

Remarque :

- *Le fonctionnement de la mémoire « RAM » et de banc des registres « Register Files » est identique à celui de TP (MRD=Mem[PC], A=RF[ay], B=RF[az]).*
- *Pour les opérations arithmétiques «+,-» les nombres binaires sont codés sur 16 bits en complément à deux.*

Le pointeur sur la mémoire du programme « RAM » doit être stocké dans une variable « *PC* » « *Program Counter* ».

```
✓ in    RF(ax), addr                /* iord = 1, address_bus = addr, RF(ax),  = data_bus */
✓ out   addr, RF(ay)                /* iowr = 1, address_bus = addr, data_bus = RF(ay)*/
✓ jmpd  addr                        /* saut inconditionnel à l'adresse addr */
✓ jmpcd RF(ay), addr                /* saut conditionnel si RF(ay) =1 à l'adresse addr */
✓ mov   RF(ax), RF(az)              /* le contenu du registre RF(az) est transféré au registre
                                       RF(ax)
✓ ldc   RF(ax) <- MEM(pc)           /* le constant stocké dans le mot suivant de la mémoire
                                       est transféré dans le registre RF(ax) */
✓ add   RF(ax) <- RF(ay) + RF(az)   /* le résultat de l'opération RF(ay) + RF(az) est
                                       transféré dans le registre RF(ax) */
✓ sub   RF(ax) <- RF(ay) - RF(az)   /* le résultat de l'opération RF(ay) - RF(az) est
                                       transféré dans le registre RF(ax) */
```

ou « *RF* » est un « *Register File* » et ax, ay, az $\in \{0, 1, .. 15\}$

Figure 1. Le jeu d'instructions du processeur.

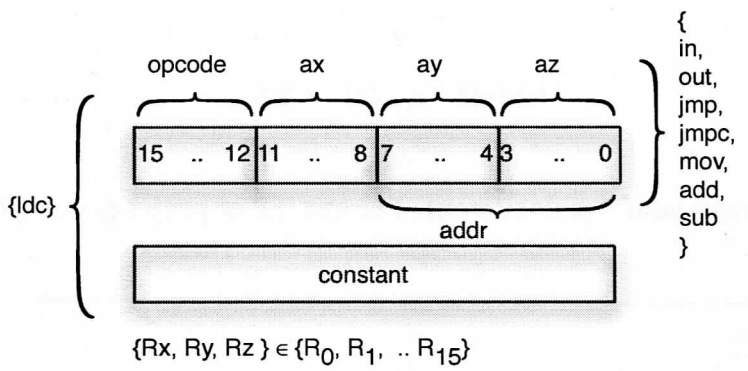

Figure 2. Composition des instructions.

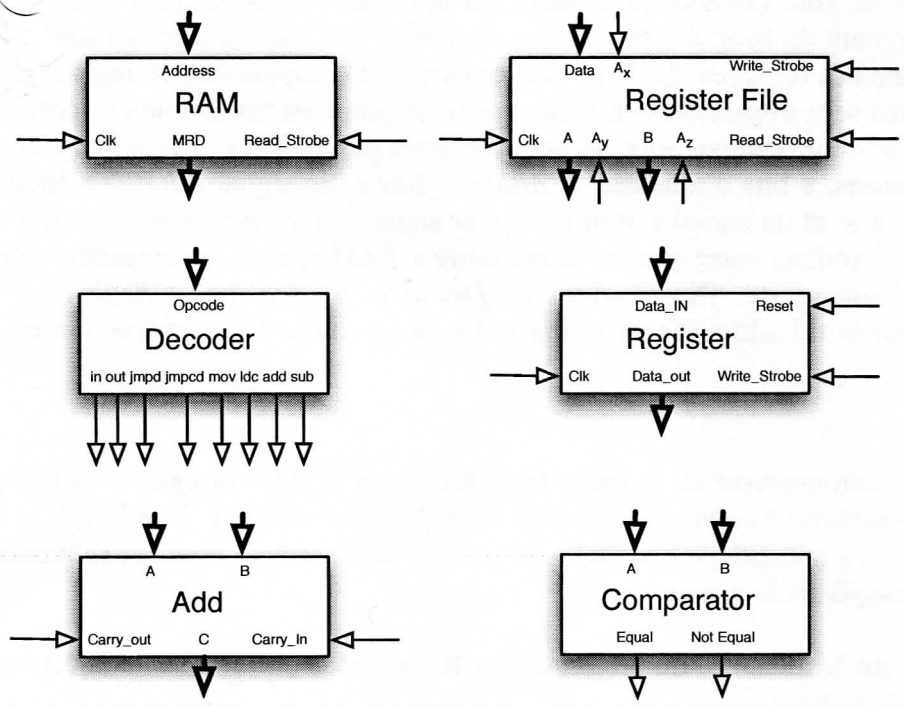

Figure 3. Les composants.

1. **(5p)** Spécifier le processeur en C* en utilisant entre autres une instruction « Switch » (Prog1) et puis une instruction « When » (Prog2) (pour simplifier l'unité de contrôle).
2. **(2p)** Dessiner le diagramme d'états complété par les affectations de variables, les calculs d'expressions et de relations correspondants.
3. **(3p)** Spécifier les affectations des variables en complétant les conditions avec des états de l'unité de contrôle.
4. **(10p)** Dessiner le schéma du processeur en utilisant entre autres les modules : « *RAM* », « *Register Files* », « *Decoder* », « *Register* », « *Add* », « *Comparator* » (Figure 3).