Supélec
2ème année Gif
Toutes voies

Architecture des systèmes informatiques

TD n°1 – Conception d'une machine à registres

1 Introduction

L'un des facteurs qui limitent la performance globale d'un système informatique est le temps de transfert des données entre le processeur et la mémoire centrale. Il existe plusieurs techniques pour améliorer ce point ; nous allons nous intéresser à l'une d'entre elles : la multiplication des registres (vous verrez, plus tard dans le cours, une autre technique : la mémoire cache).

L'idée principale qui motive l'augmentation du nombre de registres est simple : ce sont les données dans les registres qui sont le plus rapidement disponibles pour l'UAL, donc il est plus rapide de faire, par exemple, une addition entre 2 registres qu'une addition entre 1 accumulateur et un mot en mémoire centrale comme dans le cas de la machine de cours.

2 Description de l'architecture externe

Nous conserverons pour ce processeur les mêmes choix que pour la machine de cours : architecture Von Neumann, mots et adresses de 16 bits. Au lieu d'un accumulateur, 16 registres (notés R0 à R15) sont disponibles ; les échanges de données entre la mémoire centrale et le processeur se font uniquement avec les instructions LDRx et STRx ; les opérations arithmétiques prennent leurs opérandes dans des registres et stockent leur résultat dans un autre.

Le tableau suivant décrit l'architecture externe :

Instruction	Format	Description
`LOAD #val,Rx`	`0000 ----`[1] `---- xxxx`[2] `val`[3]	Chargement dans `Rx` de `val` (mode immédiat)
`LOAD adr,Rx`	`0001 ---- ---- xxxx adr`[4]	Chargement dans `Rx` du mot d'adresse `adr` (mode direct)
`LOAD (Ry),Rx`	`0011 ---- yyyy xxxx`	Chargement dans `Rx` du mot dont l'adresse est contenue dans `Ry` (mode indirect registre)
`STORE Rx,adr`	`0101 ---- ---- xxxx adr`	Rangement du contenu de `Rx` en mémoire à l'adresse `adr`
`STORE Rx,(Ry)`	`0111 ---- yyyy xxxx`	Rangement du contenu de `Rx` en mémoire à l'adresse contenue dans `Ry` (mode indirect registre)
`TFR Ry,Rx`	`0010 ---- yyyy xxxx`	Transfert de `Ry` dans `Rx`
`ADD Ry,Rz,Rx`	`1000 yyyy zzzz xxxx`	Additionne `Ry` et `Rz` et range le résultat dans `Rx`
`SUB Ry,Rz,Rx`	`1001 yyyy zzzz xxxx`	Soustrait `Rz` de `Ry` et range le résultat dans `Rx`

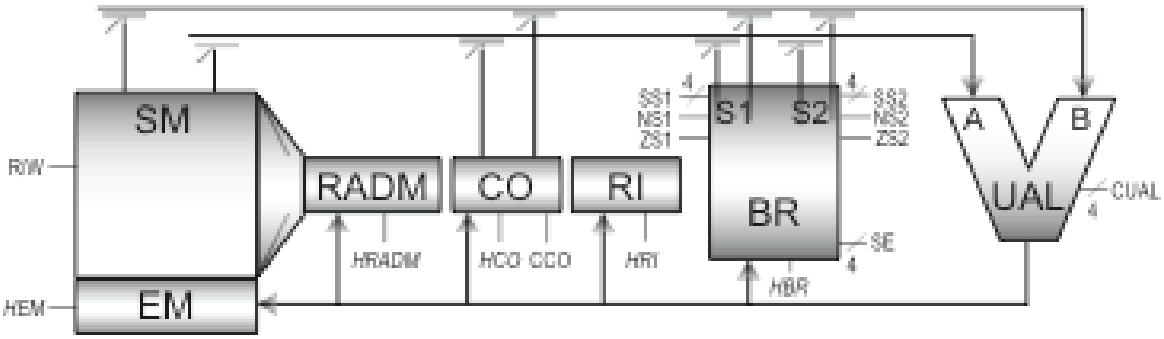
[1] ces quatre tirets signifient que ces 4 bits sont ignorés
[2] numéro du registre Rx sur 4 bits
[3] entier codé en complément à 2 sur 16 bits
[4] adresse sur 16 bits

`JMP adr`	`1100 ---- ---- ----` `adr`	Saut à l'adresse `adr`
`JZ Rx,adr`	`1110 ---- ---- xxxx` `adr`	Saut à l'adresse `adr` si le contenu de `Rx` est nul
`JN Rx,adr`	`1111 ---- ---- xxxx` `adr`	Saut à l'adresse `adr` si le contenu de `Rx` est négatif

Vous noterez que, comme pour la machine de cours, les instructions sont sur 1 ou 2 mots.

3 Chemin de données générique

Le chemin de données qui servira de base à la conception du processeur reprend celui de la machine de cours en substituant au registre accumulateur un banc de 16 registres avec 1 entrée (sélection du registre avec le signal SE, chargement sur le front montant de HBR) et 2 sorties (sélection avec les signaux SS1 et SS2) ; à chaque sortie sont associées 2 indicateurs indiquant si celle-ci est nulle (ZSi) ou négative (NSi). L'horloge du compteur ordinal (HCO) charge le CO si le signal CCO vaut 1, l'incrémente sinon. Les signaux de commande des sorties 3 états sur les bus ne sont pas nommés. L'unité arithmétique et logique est supposée posséder toutes les opérations nécessaires.



4 Travail à effectuer

Déterminer le chemin de données et les équations du séquenceur du processeur décrit.

5 Solution possible

A VERIFIER

Fetch	T1	T2
`Fetch`	CO->A->RADM CUAL=A HRADM	SM->B->RI CUAL=B HRI CCO=0, HCO

	T3	T4	T5
`LOAD #val,Rx`	CO->A->RADM CUAL=A HRADM	CCO=0, HCO	SM->B->BR CUAL=B SE=Rx, HBR
`LOAD adr,Rx`	CO->A->RADM CUAL=A HRADM	SM->B->RADM CUAL=B HRADM CCO=0, HCO	SM->B->BR CUAL=B SE=Rx, HBR

`LOAD (Ry),Rx`		S1->A->RADM CUAL=A SS1=Ry HRADM	SM->B->BR CUAL=B SE=Rx, HBR
`STORE Rx,adr`	CO->A->RADM CUAL=A HRADM	SM->B->RADM CUAL=B HRADM CCO=0, HCO	S1->A->EM CUAL=A SS1=Rx HEM, R/W=0
`STORE Rx,(Ry)`		S1->A->RADM CUAL=A SS1=Ry HRADM	S1->A->EM CUAL=A SS1=Rx HEM, R/W=0
`TFR Ry,Rx`			S1->A->BR CUAL=A SS1=Ry SE=Rx, HBR
`ADD Ry,Rz,Rx`			S1->A, S2->B CUAL=A+B SS1=Ry, SS2=RZ SE=Rx, HBR
`SUB Ry,Rz,Rx`			S1->A, S2->B CUAL=A-B SS1=Ry, SS2=RZ SE=Rx, HBR
`JMP adr`	CO->A->RADM CUAL=A HRADM	CCO=0, HCO	SM->B->CO CUAL=B HCO=1, HCO
`JZ Rx,adr`	CO->A->RADM CUAL=A HRADM	CCO=0, HCO	SS1=Rx SM->B->CO CUAL=B CCO=1, HCO si ZS1
`JN Rx,adr`	CO->A->RADM CUAL=A HRADM	CCO=0, HCO	SS1=Rx SM->B->CO CUAL=B CCO=1, HCO si NS1