INF1600

Architecture des Micro-Ordinateurs

Laboratoire #1 : Architecture du processeur

Soumis par :

Nathan, Ramsay-Veljens (1989944)

Cassy, Charles (1947025)

Section de labo #4

16 Février 2020

Exercice 1 Révision de logique et arithmétique numérique

1. **Pour les nombres entiers signés suivants, représentés en complément à deux sur le nombre de bits affichés, donnez la valeur décimale.**

   a) **10110101 (binaire) :**

   En complément à un : 10110100

   En binaire non-signé : 01001011 -> 75

   Le nombre correspondant est -75.

   b) **00110110 (binaire) :**

   Le nombre est positif, donc il ne change pas : 00110110 -> 54

   Le nombre correspondant est 54.

   c) **7027 (octal) :**

   En binaire : 1110 0001 0111 1110 0001 0111 0001 1110 1001

   En complément à un : 1110 0001 0111 1110 0001 0111 0001 1110 1000

   En binaire non-signé : 0001 1110 1000 0001 1110 1000 1110 0001 0111 -> 489.

   Le nombre correspondant est -489.

   d) **FACE (hexadécimal, 16 bits) :**

   En binaire : 1111 1010 1100 1110 0000 0101 0011 0010

   En complément à un : 1111 1010 1100 1110 0000 0101 0011 0001

   En binaire non-signé : 0000 0101 0011 0001 1111 1010 1100 1110 -> 1330

   Le nombre correspondant est -1330.

e) **1000 0001 (binaire) :**

En complément à un : 1000 0000

En binaire non-signé : 0111 1111 -> 127

Le nombre correspondant est -127.

2. **Pour chacun des 4 numéros ci-dessous (on ne connait pas la base de la représentation apriori), indiquez les bases possibles :**

ID	Numéros	BIN	OCT	DEC	HEX
(a)	817	Non	Non	Possible	Possible
(b)	A10101	Non	Non	Non	Possible
(c)	911	Non	Non	Possible	Possible
(d)	0	Possible	Possible	Possible	Possible
(e)	123	Non	Possible	Possible	Possible

3. **Expliquez dans vos mots ce que fait la ligne suivante :**
y = (x | (2 << 4)) & 254; avec x = 0x5A5A

a) Le x en binaire vaut 0101 1010 0101 1010. Le décalage vers la gauche de la valeur 2 de 4 bits vers la gauche donne 0010 000.
b) Le système fait une comparaison OU bit à bit entre ces deux valeurs précédentes. Ce qui équivaut à : 0101 1010 0111 1010.
c) Cette dernière est ensuite comparée à la valeur 254 en binaire : 1111 1110 avec un ET bit à bit qui donne la valeur finale de y : 0111 1010 qui vaut 122 en décimal.

4. **Pour les nombres entiers suivants écrits en base décimale, donnez la représentation au format binaire signé (complément à deux) 16-bit et en hexadécimal16-bit.**
a) -1999 :
(0000 0111 1100 1111)
En binaire signé : 0000 1000 0011 0001
En hexadécimal : F831

b) -32 :
(0000 0000 0010 0000)
En binaire signé :0000 0000 1110 0000

En hexadécimal : FFE0

c) 5432 :
(0001 0101 0011 1000)
En binaire signé : 0000 1010 1100 1000
En hexadécimal : 1538

5. Effectuez les opérations arithmétiques suivantes sur 12bits et indiquez s'il y a un débordement signé.

a) DAD + ACE
(1101 1010 1101) + (1010 1100 1110) = 1000 0111 1011 avec un débordement de 1
C'est un débordement signé car les deux nombres sont signés.
En hexadécimal: 87B

b) 10A + F50
(0001 0000 1010) + (1111 0101 0000) = 0000 0101 1010 avec une retenue finale de 1
Il n'y a pas de débordement car on ne peut pas avoir un débordement signé avec deux nombres de signes différents.
En hexadécimal : 05A

6. En informatique, on est souvent confronté à utiliser une organisation différente des nombres multioctets. Les deux façons courantes d'organiser les octets sont big-endian (par exemple Sun SPARC, Apple) et littleendian (x86).Un nombre entier non signé de la longueur de 4 octets (par exemple, l'adresse du premier secteur d'une partition) est stocké dans les octets oc2, oc3, oc4, oc5.

C2BB 0861 9EEC 38A0

En big-endian :
oc0 : C2
oc1 : BB
oc2 : 08
oc3: 61
oc4: 9E
oc5: EC
oc6: 38
oc7: A0
Le nombre étant compris entre oc2 et oc5, il est donc : 0861 9EEC, d'où sa valeur décimale : 140 615 404.

En little-endian :
oc0 : A0
oc1 : 38
oc2 : EC
oc3: 9E
oc4: 61
oc5: 08
oc6: BB
oc7: C2
Le nombre étant compris entre oc2 et oc5, il est donc : EC9E 6108, et sa valeur décimale : 3 969 802 504.

Exercice 2 Disque dur

Soit le tableau suivant :

Vitesse de rotation		5400 RPM
Zone	Pistes/Zone	Secteurs/Piste
1	541	783
2	937	870
3	1210	532
4	1853	841

a) La capacité ou l'espace total sur le disque dur est :
Espace total = (donnée/secteur) * ∑ ((secteur/piste) *(piste/zone))
(541*783*512) +(937*870*512) +(1210*532*512) +(841*1853*512) =
216 884 736 + 417 377 280 + 329 584 640 + 797 886 976 =
1 761 733 632B ou 1,64 GiB.

b) Le taux de lecture moyenne est :
Le taux de lecture pour une zone est :
((secteur/piste)*(rotation/seconde)*(octet/secteur)*(bits/octets))/zones
Zone 0= (5400/60*783*512*8) /(10^6)=288,65 Mb/s

Zone 1= (5400/60*870*512*8) /(10^6)= 320,72 Mb/s
Zone 2= (5400/60*532*512*8)/(10^6)= 196,12 Mb/s
Zone 3= (5400/60*841*512*8)/(10^6)= 310,03 Mb/s
Le taux de lecture moyenne pour les 4 zones:
(288,65 + 320,72+196,12+310,03) /4 = 278,88 Mb/s

c) Le disque dur a un bus PCIe de vitesse 3000 Mb/s, ce qui est nettement supérieure au taux de lecture moyenne de notre disque qui est 278,88 Mb/s, ce qui n'affecte pas la vitesse du disque. Ce qui nous permet de conclure que le taux de lecture moyenne effective est de 278,88 Mb/s.

d) Les résultats changeraient car la capacité du disque augmenterait avec un nombre supérieur de surfaces.

Exercice 3 Description RTN

1. **SUBMUL Ra, Rb,k**

   k=5
   IR <- M[PC] : PC <- PC+4;
   SUBMUL(:=op=3) -> Ra <- (Ra-Rb)/k;

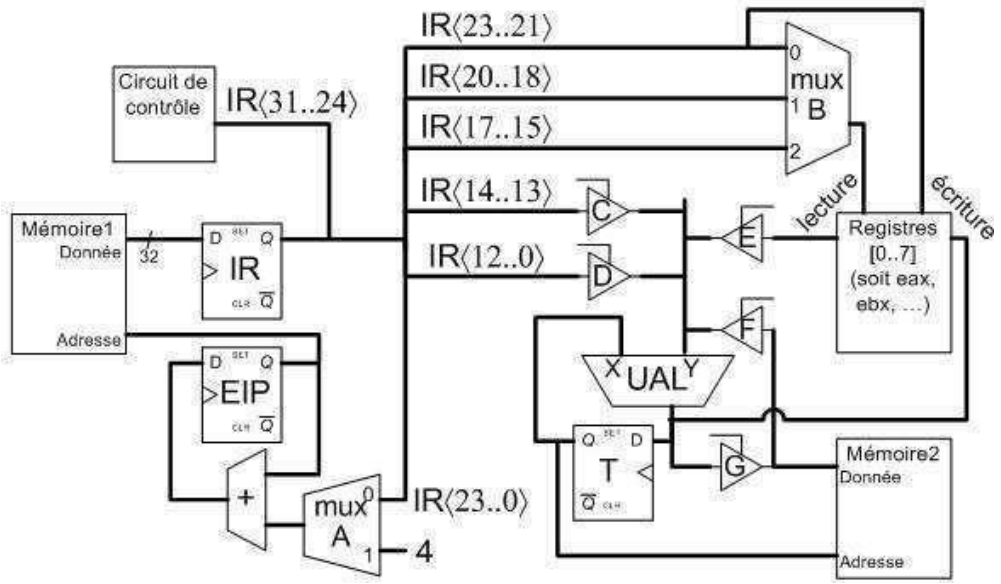
   (IR<31..27>=3)->R[IR<26..22>]<-((R[IR<26..22>-R[IR<21..17>)/(IR<16..0>));
   avec IR<16..0>=5.

2. **DECREM Ra,Rb**

   IR <- M[PC] : PC <- PC+4;
   DECREM(:=op=8) -> Rb <- (k*Rb): Ra<-(k*Ra);

   (IR<31..27>=8)->(R[IR<26..22>]<- ((IR<16..0>) * R[IR<26..22>]) : (R[IR<21..17]) <- (IR<16..0>) * R[IR<21..17>]);

Exercice 4 - Architecture d'un microprocesseur



Partie 1.

1. Soit l'instruction:

r1 <- Memoire2 [r4] << 0x16

Registres	31..24	23..21	20..18	17..15	14..13	12..0
Valeur	0000 1000	001	100	000	00	0 0000 0001 0110
	opcode	R1	R4	Inutile	Inutile	Constante = 0x16

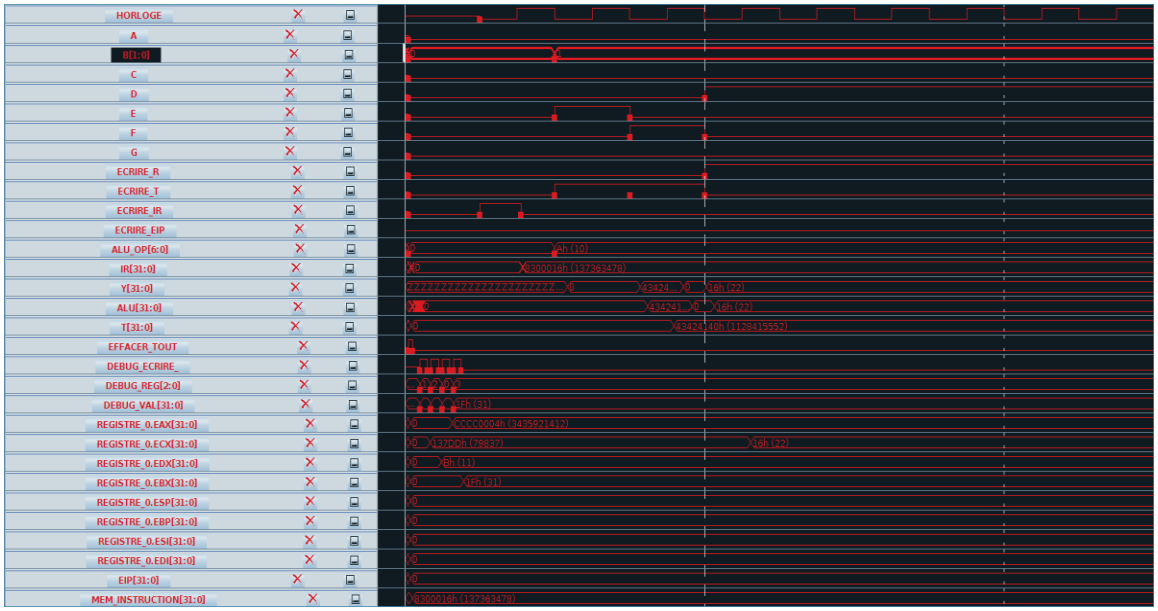
L'encodage en binaire:
0000 1000 0011 0000 0000 0000 0001 0110 -> 0x08 30 00 16 (big endian).
En little-endian: 0x 16 00 30 08.

2. Le RTN concret de l'instruction précédente est :
   T<- R[IR<20..18>];
   T<-Memoire2[T];
   R[IR<23..21>]<-T<<R[IR<12..0>];

3. **Sous forme de tableau :**

Cycles	A	B	C	D	E	F	G	UAL	ecrir eEIP	ecrir eT	ecrireRegi stre
1.	0	0b01	0	0	1	0	0	0x0A	0	1	0
2.	0	0	0	0	0	1	0	0x0A	0	1	0
3.	0	0b01	0	1	0	0	0	0x10	0	0	1

4. **Simulez l'instruction avec le logiciel Electric. Faites une ou plusieurs captures d'écran montrant clairement que le résultat de la simulation est correct, en justifiant pourquoi, et joignez-la dans le rapport suite à la question c) précédente.**



Avec les informations trouvées dans le tableau précédent nous avons utilisé les mêmes valeurs de signaux de contrôle. On peut voir qu'au premier cycle, le code de l'instruction est lu (8300016h) et stocké en mémoire, ensuite, la valeur du registre r4 est lu et mis dans T (4342140h). Dans le deuxième cycle, la valeur à l'adresse r4 de la mémoire est mis dans T. Pour le troisième cycle, le résultat est écrit dans r1 (16h), au niveau de EAX.

Partie 2.

1. Soit l'instruction :

r1 <- (Memoire2[r4] + r2) >> 0x08

Registres	31..24	23..21	20..18	17..15	14..13	12..0
Valeur	0000 0100	001	100	010	00	0 0000 0001 0110
	opcode	R1	R4	R2	Inutile	Constante=0x08

L'encodage en binaire : 0000 0100 0011 0001 0000 0000 0000 1000

En hexadécimal : 0x 04 31 00 08 (big endian)

En little endian : 0x 08 00 31 04.

2. **Le RTN concret de l'instruction précédente est :**

T <- R[IR<20...18>];

T <- Memoire2[T];

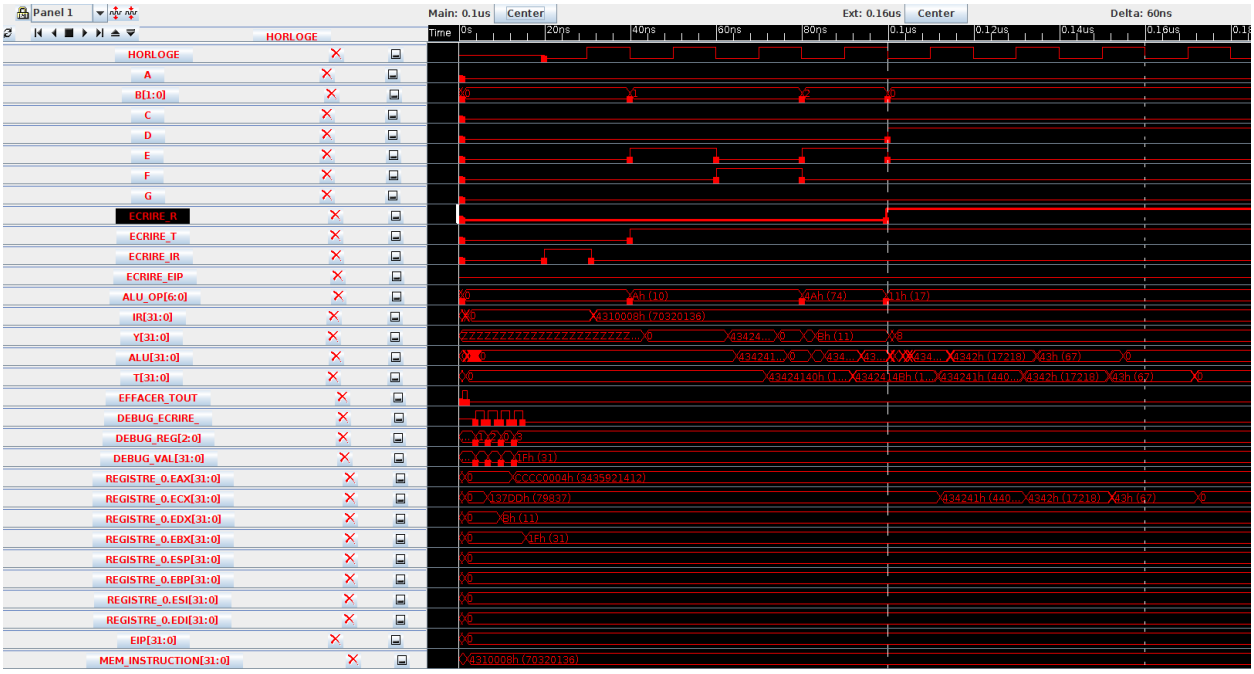
T<- T + R[IR<17..15>];

R[IR<23..21>] <- T >> IR<12..0>;

3. **Sous forme de tableau :**

	Signaux de contrôle										
Cycles	A	B	C	D	E	F	G	UAL	ecrire EIP	ecrire T	ecrireRegistre
1.	0	0b01	0	0	1	0	0	0x0A	0	1	0
2.	0	0b01	0	0	0	1	0	0x0A	0	1	0
3.	0	0b10	0	0	1	0	0	0x4A	0	1	0
4.	0	0b00	0	1	0	0	0	0x11	0	0	1

4. **Simulez l'instruction avec le logiciel Electric. Faites une ou plusieurs captures d'écran montrant clairement que le résultat de la simulation est correct, en justifiant pourquoi, et joignez-la dans le rapport suite à la question c) précédente.**



On peut voir que l'instruction entre premièrement dans la mémoire à la dernière ligne (MEM_INSTRUCTION[31 :0]) pour ensuite être chargé dans IR (IR[31 :0]). La valeur de r[4] (Bh) est ensuite lue et mise dans T. Par la suite, la valeur de Memoire2[T] (4310008h) est lue et placé dans T. On additionne par la suite r[2](Bh) et la valeur de T en écrivant le résultat dans T(434241h). En activant D, la constante 0x08 est par la suite lue pour décaler les bits du nombre dans T. Enfin, le résultat est mis dans r[1](434241h). Pour conclure, les instructions fonctionnent tel que voulu.