INF1600-Architecture des micro-ordinateurs  
 TP1 Architecture du processeur  
 Polytechnique Montréal

Nom: Georges Louis  
Matricule: 1880098

Nom: Nicolas Anid  
Matricule: 1797254  
Groupe de laboratoire: B2.  
Date de remise: Lundi 2 octobre.

Exercice 1:

1)Tous les nombres sont représentés en complément à deux.

1. 11110101 (binaire)  
   -128+64+32+16+4+1 = -11 (décimale)
2. 00101010 (binaire)  
   2+8+32 = 42 (décimale)
3. 4517 (octal)

100 101 001 111 (binaire)

-2048+256+64+8+4+2+1= -1713 (décimale)

d) CAFE (hexadécimal, 16 bits)

1100 1010 1111 1110 (binaire)

-32768+16384+2048+512+128+2+4+8+16+32+64 = -13570 (décimale)

e) 10000000 (binaire)  
 -0 = 0 (décimale)

2) Les bases possibles pour chaque cas est cochez par un ‘X’

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| ID | Numéros | BIN | OCT | DEC | HEX |
| (a) | 5781 |  |  | X | X |
| (b) | 10000000 | X | X | X | X |
| (c) | 1600 |  | X | X | X |
| (d) | B747 |  |  |  | X |
| (e) | 00000000 | X | X | X | X |

3) y = x & (3 << 4);

Le "3" en binaire "011" va être décaler de 4 bits vers la gauche puis chaque bits de numéro obtenu en binaire, va être compare a chaque bits de x de même poids. Si les deux bits sont « 1 » sa retourne « 1 ».

Exemple : 110000 & 011110 = 010000  
note : le 110000 est le « 3 » décaler de 4 bits en binaire.

4)

a) -1234

= 1111 1011 0010 1110 (binaire complément à deux)  
= FB2E (hexadécimale)

b) 32767

= 0111 1111 1111 1111(binaire complément à deux)

= 7FFF

c) -32

= 1111 1111 1110 0000(binaire complément à deux)

= FFE0

5)a)  
 7C + 4F

0111 1100  
+  
0100 1111  
-------------  
1100 1011

C B

Débordement, 2 nombres positifs qui donnent un nombre négatif.

b)  
89 + 11

1000 1001  
+  
0001 0001  
--------------  
1001 1010

9 A

Pas de débordement.

6)

oc2 = C2 qui est 1100 0010 en binaire.

oc3 = BB qui est 1011 1011 en binaire.

oc4 = 38 qui est 0011 1000 en binaire.

oc5 = A0 qui est 1010 0000 en binaire.

1. Big-endian

oc2oc3oc4oc5  
1100 0010 1011 1011 0011 1000 1010 0000 (binaire)  
=3267049632 (décimale)  
=C2BB38A0 (hexadécimale)

1. Little-endian

oc5oc4oc3oc2

1010 0000 0011 1000 1011 1011 1100 0010(binaire)  
=2688072642(décimale)

=A038BBC2(hexadécimale)

Exercice2 :

1. Zone 1 : (792\*512\*8\*624)/(2^20) = 1930.5 Mb  
   Zone 2: (1424\*780\*512\*8)/(2^20) = 4338.75 Mb  
   Zone 3: (1680\*760\*512\*8)/ (2^20) = 4987.5 Mb  
   Zone 4 : (1815\*720\*512\*8)/(2^20) = 5104.6875 Mb

La somme donne 16361.4375 Mb qui est l’espace total sur le disque dur.

1. Zone 1 : (90\*792\*512\*8)/(2^20) = 278.4375 Mb/s

Zone 2 : (90\*780\*512\*8) /(2^20) = 274.21875 Mb/s

Zone 3 : (90\*760\*512\*8) /(2^20) = 267.1875 Mb/s

Zone 4 : (90\*720\*512\*8) /(2^20) = 253.125 Mb/s  
La moyenne est égale à 268.2422 Mb/s

1. Le taux de lecture moyenne effective est celui du disque dur calculé dans la partie “b”, la vitesse de lecture du disque dur connecté avec un bus PCIe de vitesse 4000Mb/s sera limité par la vitesse du disque dur qui est plus lent et de 268.2422 Mb/s.
2. Non, parce que tout les disques sont placés en parallèles et on effectue la lecture en faisant une rotation, donc la piste à parcourire est la même.

Exercice 3 :

1. SUBMUL := IR<31..27>  
   Ra := R[IR<26..22>]  
   Rb := R[IR<21..17>]

K := R[IR<16..0>]

(SUBMUL = 5) -> R[a] <- (R[a] – R[b]) \* 8 avec K = 0x0008

1. (SUBMUL = 13) –> R[a] <- (R[a] – 1) : R[b] <- (R[b] – 1)

Exercise 4:

1. r1 <- Memoire2 [r3] +r3

On a 32 bits, 8 bits pour l’opcode, 3 bits pour chaqu’un de r1, r2, r3, r4 et r5 et le reste de bits on les a mis à 0.

1. Opcode = 1000 0000; r1 = 001; r2=010; r3=011; r4 = 000;r5 = 000;

Encodage : 1000 0000 0010 1001 1000 0000 0000 0000

En hexadécimale l’encodage sera 80298000

1. T <- EBX;

ECX <- Mémoire[2] + T;

c)

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| A | B | C | D | E | F | G | UAL | ecrireEIP | ecrireT | ecrireRegistre |
| X | 10 | 0 | 0 | 1 | 0 | 0 | 0x0a | 0 | 1 | 0 |
| X | X | 0 | 0 | 0 | 1 | 0 | 0x4a | 0 | 0 | 1 |

d)



1. r1 <- (Memoire2 [r3] +0x23) >> r2

On a 32 bits, 8 bits pour l’opcode, 3 bits pour chaqu’un de r1, r2, r3, r4 et r5 et le reste de bits on les a mis à 0.

1. Opcode= 0000 0001; r1 = 001; r2=010; r3 = 011; r4 = 000; r5 = 0x23 qui est 0010 0011 en binaire.

Encodage: 0000 0001 0010 1001 1000 0100 0110 0000

En hexadécimale l’encodage sera 01298460

b)

T <- EBX;

ECX <- Mémoire2[T];

T <- ECX + 0x23;

ECX <- T >> EDX;

c)

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| A | B | C | D | E | F | G | UAL | ecrireEIP | ecrireT | ecrireRegistre |
| X | 10 | 0 | 0 | 1 | 0 | 0 | 0x0a | 0 | 1 | 0 |
| X | X | 0 | 0 | 0 | 1 | 0 | 0x0a | 0 | 0 | 1 |
| X | X | 0 | 1 | 0 | 0 | 0 | 0x4a | 0 | 1 | 0 |
| X | X | 0 | 0 | 1 | 0 | 0 | 0x11 | 0 | 0 | 1 |

d)

