Université d'Alger 1 — Solution-Examen UEF31-ADO

10 Mars 2018

Nom : Prénom : Matricule :		/ 40

Exercice 1 : (8 points = 8 x 1 point pour chaque case correcte)

Trouvez la description la plus adéquate de chaque vocabulaire. Faites correspondre les chiffres à gauche aux lettres à droite pour les réponses. Chaque réponse doit être utilisée qu'une seule fois.

1. Unité de contrôle	a) Code binaire
2. Bit	b) Demande à la machine d'exécuter des opérations plus complexes.
3. Compteur ordinal	c) Décode les instructions et déclenche les cycles nécessaires à leur lecture et à leur exécution.
4. Mémoire	d) Contiennent les données sur lesquelles opèrent les instructions actuellement exécutées par la machine.
5. Assembleur	e) Binary digit (élément binaire qui prendre deux états 0 ou 1).
6. Registres	f) Compteur d'adresse des instructions du programme
7. Instruction	g) Composants de l'ordinateur où tous les programmes en cours d'exécution et les données associées y résident.
8. Langage machine	h) Programme symbolique qui se traduit par une suite binaire appelée code machine.

1. c	2. e	3. f	4. g	5. h	6. d	7. b	8. a

Exercice 2 : (20 points)

Voici un fragment de code en C

Version C

```
z=0;

y = y << 16;

for(i=0;i<16;i=i++)
{
  y = y >> 1;
  z = z << 1;

  if(x >= y)
  {
    x = x - y;
    z = z + 1;
  }
}
```

Version MIPS

```
# i <-> $t0, x <->$t1, y <-> $t2, z <-> $t3
# $t4 <- 16

        li   $t3 , 0
        sll  $t2, $t2, 16
        li   $t0 , 0
        li   $t4 , 16

L1:     bge  $t0 , $t4 , L2
        srl  $t2 , $t2 , 1
        sll  $t3 , $t3 , 1

        blt  $t1 , $t2 , FI
        sub  $t1 , $t1 , $t2
        addi $t3 , $t3 , 1
FI:
        addi $t0, $t0, 1
        j L1
L2:
```

1/ Traduisez ce fragment de code dans le langage assembleur du MIPS. (6 points)

2/ Que fait ce programme ? (4 points)

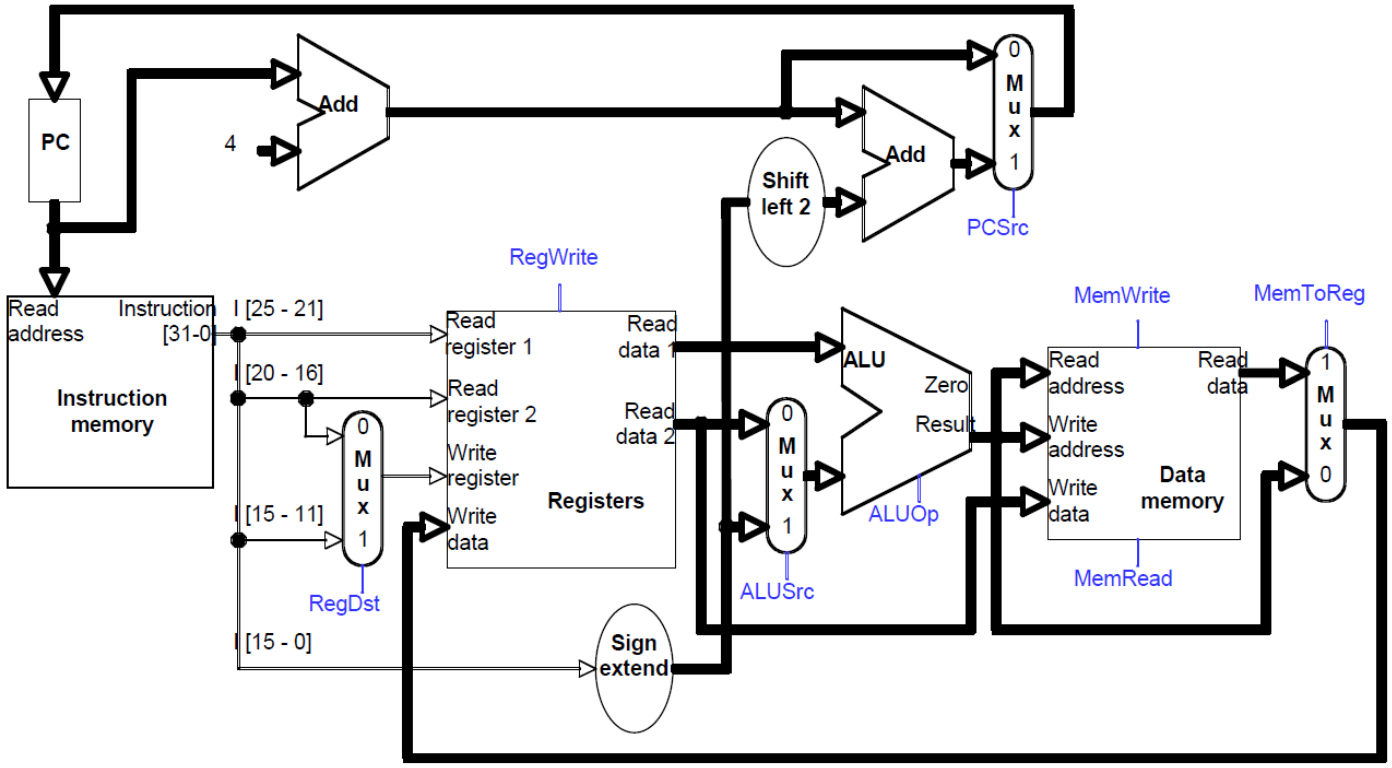
La division (2 points) z = x / y (1 point). Le reste de la division est stocké dans x (1 point).

3/ Proposer une version simplifiée. (10 points)

```
      # x <->$t1, y <-> $t2, z <-> $t3

(6 points)  divu* $t1, $t2     # x / y.
                               # LO contient le quotient de la
                               # division x / y.

                               # HI contient le reste de la
                               # division x / y

(2 points)  mflo $t3     # $t3 <- quotient de la division x / y.
(2 points)  mfhi $t1     # $t1 <- reste de la division x / y.

*On peut déduire de la version C que x et y sont des valeurs sur
16 bits et de ce fait l'instruction de division signée « div » est
aussi une réponse acceptable.
```

Exercice 3 : (12 points = 24 x 0,5 point pour chaque case correcte)

Soit le chemin de donnée (datapath) à cycle-unique suivant



Remplir les blancs dans le tableau ci-dessous avec les valeurs des signaux de contrôle pour exécuter les instructions indiquées. Chaque signal de contrôle doit être spécifié comme 0, 1 ou X. Ecrire un 1 ou un 0 quand un X est plus précis n'est pas correct. Si un signal de contrôle est une fonction logique d'un ou de plusieurs autres signaux, écrivez cette fonction. Pour ALUOp, choisissez entre ("and", "or", "add", "sub" ou "slt").

OpCode	RegDst	RegWrite	ALUSrc	ALUOp	MemWrite	MemRead	MemToReg	PCSrc
sub	1	1	0	sub	0	0	0	0
lw	0	1	1	add	0	1	1	0
bne	X	0	0	sub	0	0	X	!zero && bne