

Nom :

Prénom :

N° Étudiant :

Groupe de TD :

Partiel 2021 – 2022
Architecture des ordinateurs 1 – LU3IN029
Durée : 1h30

Documents autorisés : Aucun document ni machine électronique n'est autorisé à l'exception du mémento MIPS.

Le sujet comporte 11 pages. Ne pas désagrafer les feuilles. Répondre directement sur le sujet. Le barème indiqué pour chaque question n'est donné qu'à titre indicatif. Le barème total est lui aussi donné à titre indicatif : 33 points.

Le partiel est composé d'exercices indépendants.

- Exercice 1 - 11 points : Arithmétique saturée – (p. 1)
- Exercice 2 - 11 points : Données et mémoire – (p. 6)
- Exercice 3 - 11 points : Inversion en place d'une chaîne – (p. 9)

Exercice 1 : Arithmétique entière et saturée – 11 points

Dans cet exercice, on considère un additionneur 8 bits, que l'on appelle ADD8.

Il a deux entrées $A = a_7a_6a_5a_4a_3a_2a_1a_0$ et $B = b_7b_6b_5b_4b_3b_2b_1b_0$ qui sont des mots binaires de 8 bits.

Sa sortie est composée du mot binaire de 8 bits $S = s_7s_6s_5s_4s_3s_2s_1s_0$ et du drapeau OV permettant de détecter un dépassement de capacité sur entiers relatifs. Ce drapeau est calculé par l'expression $cout_7 \oplus cout_6$ avec $cout_i$ la retenue du rang sortante du rang i .

Question 1.1 : 1 point

On souhaite réaliser l'addition entre les valeurs décimales 117 et -38. Donner les valeurs à mettre en entrée de ADD8 (soit les valeurs des entrées A et B) pour réaliser cette opération. L'addition de ces deux entrées génère-t-elle un dépassement de capacité sur entiers relatifs ? Justifier toutes les réponses.

Remarque : le résultat de l'addition n'est pas demandé.

Question 1.2 : 1 point

Il est possible de détecter un dépassement de capacité sur entiers relatifs uniquement à partir (de certains bits) des entrées A et B ainsi que (des bits) de la sortie S. On note OV2 le drapeau permettant cette detection avec un calcul différent de celui de OV.

Donner une expression booléenne de OV2 et une phrase rédigée expliquant la réponse.

Question 1.3 : 3 points

On considère maintenant les deux mots binaires $m_1 = 0b01010010$ et $m_2 = 0b01100100$.

Réaliser ci-dessous l'addition de m_1 et m_2 . Quelles sont les sorties S et OV de l'additionneur ADD8 ?

Si ces deux mots représentent des entiers relatifs codés en complément à deux, quelles valeurs décimales additionne-t-on ? L'opération génère-t-elle un dépassement de capacité ? Justifiez vos réponses.

Si ces deux mots représentent des entiers naturels codés en base 2, quelles valeurs décimales additionne-t-on ? Quelles sont les sorties S et OV de l'additionneur ADD8 ? Cette opération génère-t-elle un dépassement de capacité sur entiers naturels ? Comment peut-on le détecter ? Justifiez vos réponses.

On souhaite réaliser à partir de l'additionneur 8 bits ADD8 décrit ci-dessus un additionneur 8 bits à saturation nommé ADD8SAT. Cet additionneur à saturation est dédié aux additions d'entiers relatifs. En cas de dépassement de capacité (sur entiers relatifs donc), il renvoie la valeur représentable la plus proche du résultat correct.

ADD8SAT, comme ADD8, a deux entrées A et B sur 8 bits. Il a comme sortie un mot binaire de 8 bits que l'on note $R = r_7r_6r_5r_4r_3r_2r_1r_0$.

Le fonctionnement de l'additionneur à saturation ADD8SAT est le suivant. Les deux entrées de ADD8SAT sont passées à ADD8 pour réaliser une addition. Lorsque cette addition avec ADD8 ne génère pas de dépassement de capacité sur entiers relatifs alors la sortie R est identique à la sortie S de ADD8.

Lorsque l'addition avec ADD8 génère un dépassement de capacité sur entiers relatifs alors :

- si A et B représentent des entiers négatifs, le résultat R vaut la plus petite valeur représentable en complément à deux sur 8 bits ;
- si A et B représentent des entiers positifs, le résultat R vaut la plus grande valeur représentable en complément à deux sur 8 bits.

Question 1.4 : 1 point

Sur 8 bits, en complément à deux, quelle est la plus petite valeur représentable ? Donner sa valeur décimale et sa représentation binaire.

Sur 8 bits, en complément à deux, quelle est la plus grande valeur représentable ? Donner sa valeur décimale et sa représentation binaire.

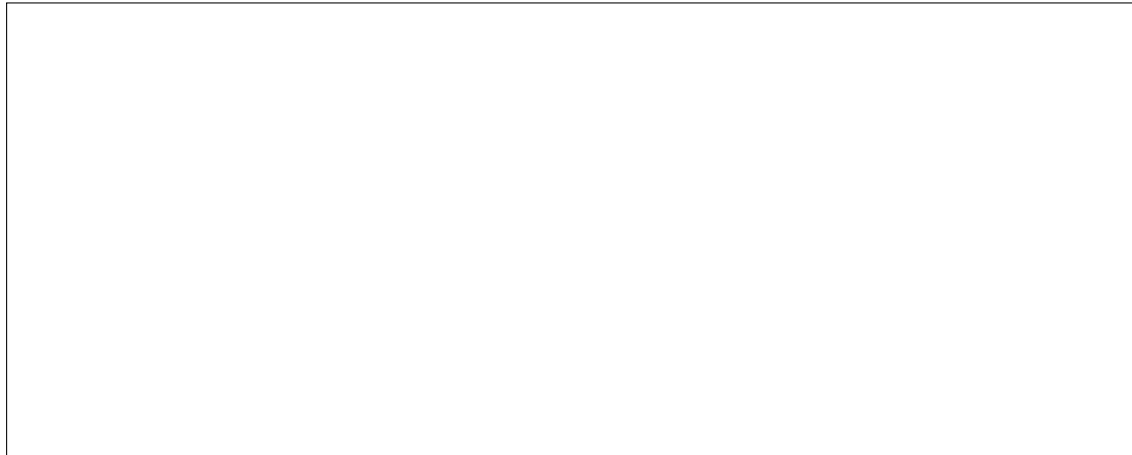
Question 1.5 : 3 points

Donner une expression booléenne de r_7 , le bit de poids fort du résultat R de ADD8SAT, en fonction de certains bits des entrées A et de B ainsi que de certains des bits des sorties (S et OV) de ADD8.

Expliquer avec une phrase rédigée votre réponse.

Donner une expression booléenne des bits r_i avec $i \in [6, 0]$, les bits de rang 6 à 0 du résultat R de ADD8SAT, en fonction de certains bits des entrées A et de B ainsi que de certains des bits des sorties (S et OV) de ADD8.

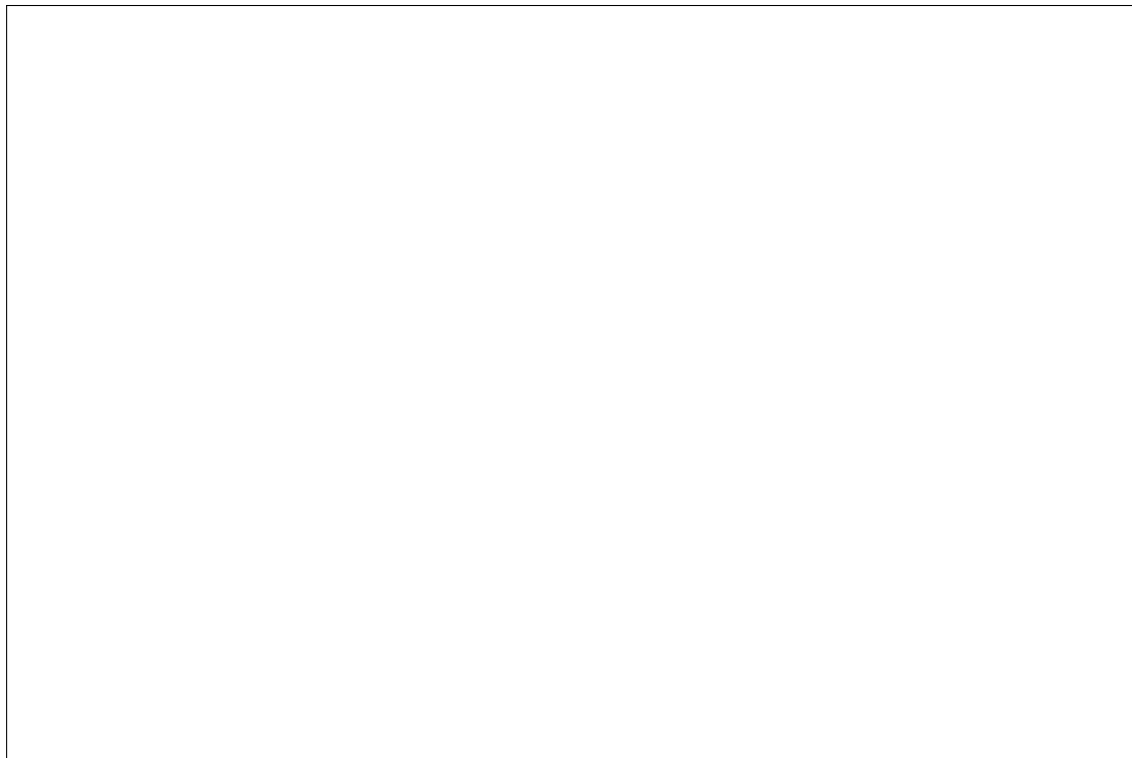
Expliquer avec une phrase rédigée votre réponse.



Question 1.6 : 2 points

Donner le codage binaire de l'instruction d'addition sans détection de dépassement de capacité suivante :
addiu \$18, \$13, -5 en justifiant votre réponse.

Donner aussi la représentation hexadécimale de l'instruction.



Exercice 2 : Données et mémoire – 11 points

On considère le programme C suivant :

```
signed char  a      = 0x2;
signed short b      = 0x0014;
int  tab[]  = {0xFFFFFFFF, 0xFFFFFFFFC};
signed char  c      = -1;

int main() {
    signed int  tmp1 = tab[1];
    signed char tmp2 = a;
    signed char tmp3 = c;

    tmp3 = tmp3 + tmp1 + tmp2;
    b = b + tmp3;

    printf("%d",b); /* affichage en décimal de la valeur de b */
    exit();
}
```

Question 2.1 : 3 points

Donner la section `.data` du code assembleur correspondant au programme principal ci-dessus.

Pour chacune des variables, donner sous forme de commentaire dans la section `data`, son adresse d'implantation en la justifiant.

Dans le tableau ci-dessous, donner le contenu du segment de données (pour les adresses indiquées et en hexadécimal) juste après le chargement du programme en mémoire. Toute valeur non connue sera indiquée par ??.

	Vue par octet				Vue par mot
Adresse (de mot)	+ 0	+ 1	+ 2	+3	
0x10010000					
0x10010004					
0x10010008					
0x1001000c					

Question 2.2 : 6 points

Donner la section `.text` du code assembleur **non optimisé** correspondant au programme principal ci-dessus.

Donner des commentaires permettant d'indiquer les liens entre le code C et le code assembleur. Toute allocation en pile devra être assortie d'une justification du nombre d'octets alloués.

Question 2.3 : 2 points

Quelle est la valeur affichée par le programme ?

Exercice 3 : Inversion en place d'une chaîne de caractères – 11 points

Question 3.1 : 11 points

Soit le code C suivant, qui étant donnée une chaîne de caractères globale, l'affiche puis renverse ses caractères en place (dans la même chaîne) avant d'afficher la chaîne résultante. Le nombre d'inversions réalisé est affiché à la fin du programme.

```
int size = 9;
unsigned char ch[] = "inversion";

void main() {
    int i = 0;
    int j = size - 1;
    unsigned char tmp;

    printf("%s", ch);

    while (i < j) {
        tmp = ch[j];
        ch[j] = ch[i];
        ch[i] = tmp;
        i++;
        j--;
    }

    printf("%s", ch);
    printf("%d", i);
    exit();
}
```

Donner le code assembleur correspondant au programme ci-dessus. Le programme assembleur doit évidemment respecter les conventions d'utilisations des registres. De plus, le programme pourra être optimisé. En particulier, les variables locales pourront être optimisées en registre (mais allouées).

Votre programme doit être "fidèle" au programme C : les données doivent être allouées dans le même ordre que dans le code C. De même, les traitements (restant après optimisation) doivent être réalisés dans le même ordre en assembleur et en C.

Vous devez commenter un minimum votre code assembleur afin d'indiquer ce que réalise chaque instruction (lien avec le code C ou convention MIPS).

Tout code illisible et non commenté recevra la note 0.

