Nom :	Prénom :
N° Étudiant :	Groupe de TD :

Partiel 2022 – 2023

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 : 39 points.

Le partiel est composé d'exercices indépendants.

Exercice 1 : Instruction d'accès mémoire et codage binaire – 8 points

On s'intéresse à l'introduction d'une nouvelle instruction dans le jeu d'instructions MIPS : `lws`. Sa syntaxe asssembleur est :

```
lws Rdst, (Rsrc1, Rsrc2 lsl Imm)
```

Cette instruction effectue une lecture mémoire d'un mot (4 octets) à l'adresse `A = Rsrc1 + (Rsrc2 « Imm)`. Le signe << désigne l'opération de décalage à gauche et l'immédiat `Imm` a une valeur comprise entre 0 et 31 inclus. Ainsi, l'adresse accédée résulte de l'addition du contenu de `Rsrc1` avec le contenu de `Rsrc2` préalablement décalé à gauche de `Imm`. Le mot lu est placé dans le registre `Rdst`.

Question 1.1 : 2 points

On suppose que le contenu de la mémoire est le suivant :

	Vue par octet				Vue par mot
Adresse (de mot)	+ 0	+ 1	+ 2	+3	
0x10010004	0x01	0x02	0x03	0x04	
0x10010008					0x05060708
0x1001000c					0x10F0E0D0
0x10010010	0x14	0x13	0x12	0x11	

Remplir les cases vides du tableau ci-dessus.

En supposant que le registre `$4` contient `0x10010000` et `$6` contient `0x00000004`, quelle est la valeur contenue dans `$7` après l'exécution de l'instruction `lws $7, ($4, $6 lsl 2)` ? Justifier votre réponse.

Question 1.2 : 3 points

Combien d'opérandes a cette instruction ?

Pour chacun des opérandes, indiquer sa nature (dans l'instruction assembleur) et le nombre de valeurs qu'il peut prendre. En déduire le nombre minimal de bits nécessaires au codage de chacun de ces opérandes dans l'instruction binaire associée à l'instruction assembleur.

Existe-t-il un format de codage MIPS adapté au codage de cette instruction ? Justifier votre réponse.

Question 1.3 : 2 points

L'opération indiquée par une instruction est associée à un code de 6 bits qui est une entrée soit dans la table OPCOD, soit dans la table SPECIAL (voir le mémento MIPS).

On suppose que pour l'instruction `lws` ce code sur 6 bits est `0b111111`.

En utilisant votre réponse à la question précédente, encoder en binaire l'instruction `lws $17, ($13, $10 lsl 4)` en justifiant votre réponse.

Enfin donner le codage de cette instruction sous forme hexadécimale.

Question 1.4 : 1 point

Donner une suite d'instructions MIPS équivalente à l'instruction `lws $17, ($13, $10 lsl 4)`

Exercice 2 : Programmation assembleur – 17 points

Question 2.1 : 12 points

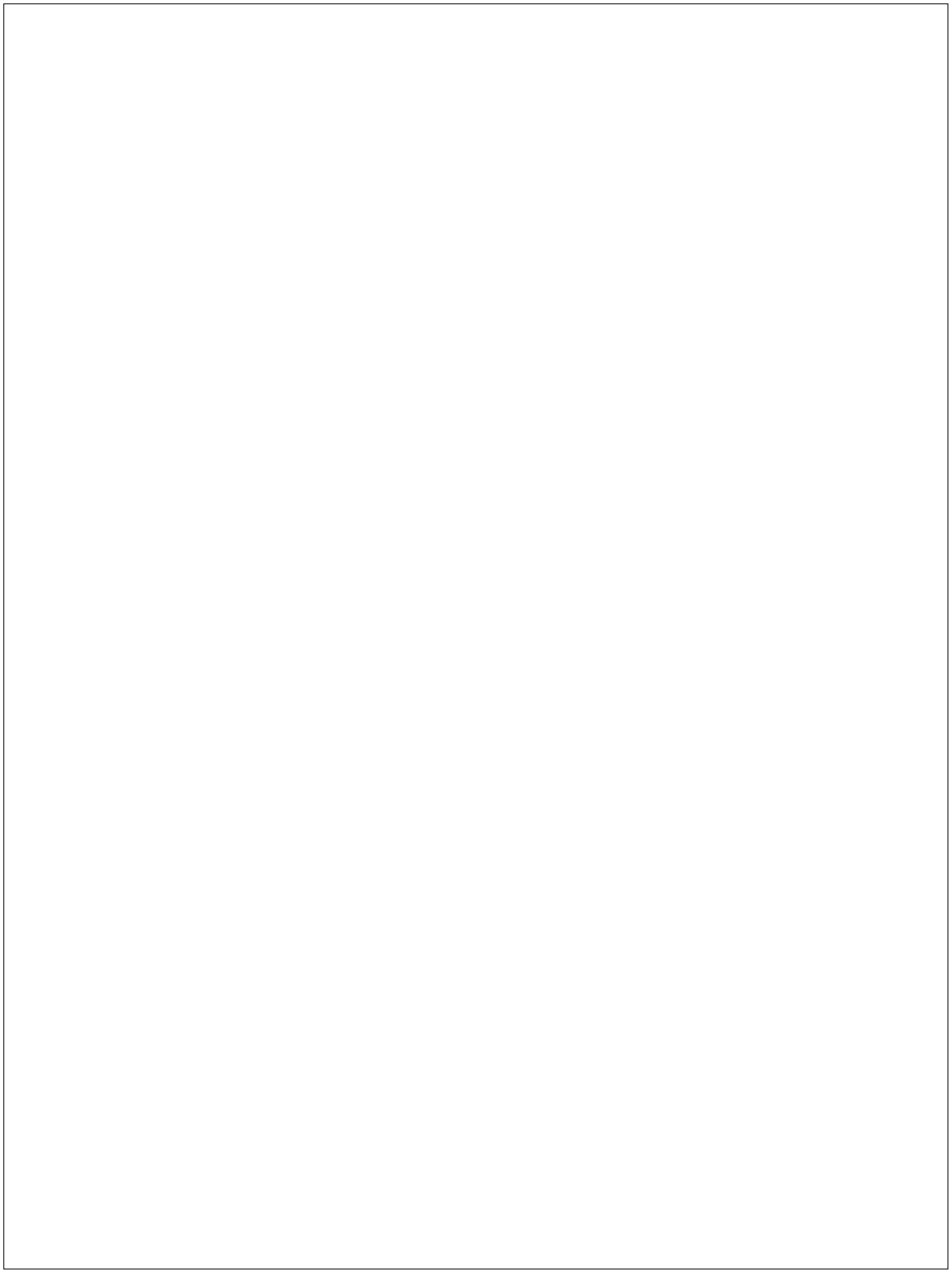
Soit le code C suivant, qui étant donné un tableau d'entiers `tab` de taille `size` (qui vaut au plus 255 car de type `unsigned char`), compte le nombre d'éléments de `tab` qui sont impairs et le nombre d'élements de `tab` qui sont pairs. Ces nombres sont affichés à la fin du programme.

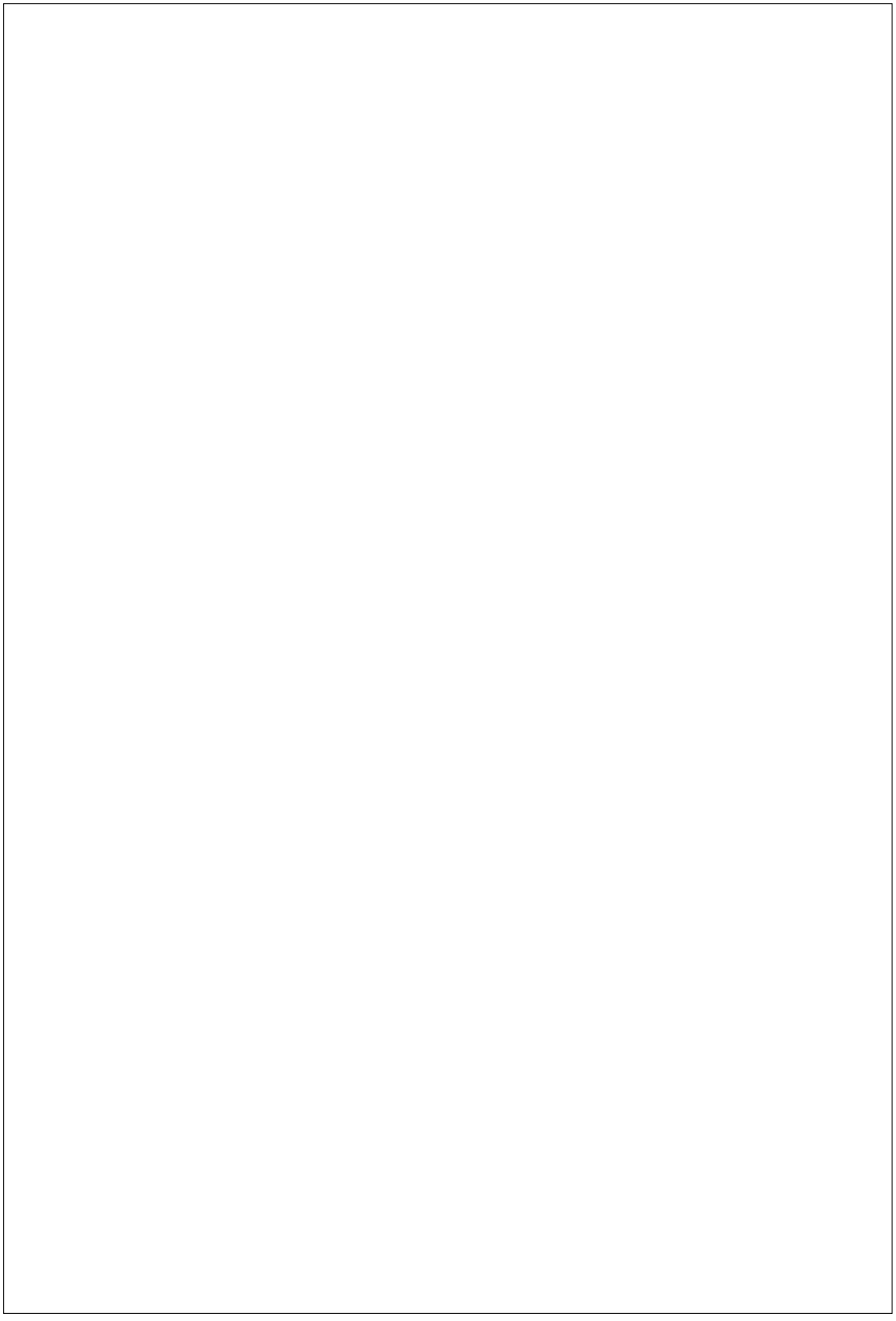
```
unsigned char size = 8;
int tab[] = {3, 5, 8, 10, 7, 25, 31, 24};

void main() {
  unsigned char i = 0;
  unsigned char nb_impair = 0;
  unsigned char nb_pair = 0;

  while (i < size)
  {
    if ((tab[i] & 0x1) == 0) /* test parité */
    {
      nb_pair++;
    }
    else
    {
      nb_impair++;
    }
    i++;
  }

  printf("%d", nb_pair);    /* affichage d'un entier */
  printf("%d", nb_impair);  /* affichage d'un entier */
  exit();
}
```

Donner le code assembleur correspondant au programme ci-dessus. Le programme assembleur NE DOIT PAS être optimisé. Vous traduirez donc de manière fidèle le programme C en assembleur.

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 ou non commenté recevra la note 0.





Question 2.2 : 5 points

On s'intéresse à l'optimisation du code donné en réponse à la question précédente.

Pour cela, numéroter les instructions de votre code réponse de la question précédente.

Donner les optimisations possibles de votre code : pour chaque optimisation possible, indiquer la ou les lignes concernées par cette optimisation. Expliquer ce qu'il faudrait changer au code pour mettre en oeuvre cette optimisation, ainsi que, le cas échéant les conséquences sur d'autres lignes du code.

Exercice 3 : Circuits logiques et ou-exclusif – 14 points

L'opération OU-EXCLUSIF sur n entrées a deux versions :

1. Le résultat vaut 1 si et seulement si une seule des n entrées vaut 1. On appelle `XOR1` cette version.
2. Le résultat vaut 1 si un nombre impair d'entrées vaut 1. On appelle `XORimpair` cette version.

On s'intéresse au cas où `n` vaut 4. On appelle `a`, `b`, `c`, `d` les 4 entrées sur 1 bit.

Question 3.1 : 3 points

Combien de lignes les tables de vérités des fonctions booléennes `XOR1` et `XORimpair` à 4 entrées (sur 1 bit) ont-elles ? Donner ces tables de vérité.

Donner la forme normale disjonctive de `XOR1` et `XORimpair`

Question 3.2 : 3 points

Simplifier les expressions obtenues à la question précédente pour faire apparaître le maximum de portes `XOR` à deux entrées dans les deux expressions. Vous détaillerez vos simplifications que vous justifierez.

On souhaite réaliser un circuit `XOR_4` qui, à partir d'une commande `cmd` et 4 entrées `a`, `b`, `c` et `d` sur 1 bits, réalise le `XOR1` si `cmd` = 0, et réalise le `XORimpair` si `cmd` = 1.

Question 3.3 : 1 point

Avec quel circuit peut on réaliser la sélection d'une valeur parmi deux ? Donner une expression booléenne de ce circuit.

Question 3.4 : 4 points

Donner une réalisation schématique du circuit `XOR_4` permettant de réaliser la sélection d'une des deux versions du `XOR` à 4 entrées. Votre réalisation doit être "optimisée" : aucun calcul ne doit être réalisé deux fois. Votre circuit ne doit contenir que des portes NOT, XOR, ET et OU. Vous pouvez utiliser des portes ET et OU avec des entrées ou sorties complémentées (mais elles doivent être dessinées de façon lisible).

Le chemin critique d'un circuit correspond au plus long chemin en nombre de portes ET-OU entre une entrée et une sortie du circuit. La taille de ce chemin est le nombre de portes ET-OU. Quel est le chemin critique de votre circuit et sa taille ? Justifier votre réponse.

Question 3.5 : 3 points

On s'intéresse désormais uniquement au `XORimpair`, plus précisément au circuit qui à partir d'un mot binaire de $n = 2^k$ bits, avec $k \geq 1$, a pour sortie 1 si le nombre de bits à 1 dans le mot est impair, 0 si ce nombre est pair.

Dans le cas où $n = 2^3$, combien de portes `xor` contient le circuit ? Justifier.

Dans le cas général où $n = 2^k$, combien de portes `xor` contient ce circuit ? Comment réduire le chemin critique (le plus long en nombre de portes ET-OU) de ce calcul ? Quel est dans ce cas sa taille ?