Examen du cours de Fonctionnement des Ordinateurs
1ère Session, Juin 2017
Partie 1

NOM : ..... PRENOM : ..... SECTION : .....

Consignes à lire impérativement !

L'examen est composé de **2 parties**. Chaque partie dure **2 heures**. Il vous est demandé de respecter les consignes suivantes.

- Commencez par écrire vos **nom**, **prénom** et **section** (math, info, ...) sur chaque feuille, y compris les feuilles de brouillon.
- Laissez vos calculatrice, téléphone portable et notes de cours dans votre sac. Leur usage n'est **pas autorisé**. Pensez à éteindre votre téléphone portable !
- Faites attention à la clarté et à l'organisation de vos réponses. Respectez les règles grammaticales et orthographiques.
- Utilisez pour vos réponses les **cadres** prévus à cet effet. Si davantage d'espace est nécessaire, utilisez le dos de la feuille ou une feuille supplémentaire et indiquez clairement où se situe le restant de la réponse.
- Vous devez terminer cette partie de l'examen avant de pouvoir sortir de la salle (pour aller à la toilette par exemple).
- Toutes les feuilles (énoncé et brouillon) doivent être remises en fin d'examen.
- Vérifiez que vous avez répondu à toutes les questions (il y a **2 questions** dans cette partie).

Question 1 – Recherche dichotomique (/6)

L'objectif de cette question est l'implémentation en langage d'assemblage MIPS d'une fonction qui effectue une recherche dichotomique dans un tableau trié d'entiers. La recherche est effectuée de façon **récursive**. Un exemple d'implémentation en Java d'une telle fonction est fourni ci-dessous. La fonction retourne -1 si le nombre recherché n'a pas été trouvé. Elle retourne l'index du nombre dans le tableau s'il a été trouvé.

```
public static int recherche(int [] tab, int bas, int haut, int c) {
  if (bas > haut)
    return -1;
  int milieu= bas+(haut-bas)/2;
  if (c == tab[milieu])
    return milieu;
  else if (c < tab[milieu])
    return recherche(tab, bas, milieu-1, c);
  else
    return recherche(tab, milieu+1, haut, c);
}
```

Dans cet exercice, la recherche dichotomique sera appliquée à un tableau constant d'entiers, stocké en mémoire comme suit : chaque entier occupe 32 bits et l'adresse de la première cellule du tableau est alignée sur 32 bits. Pour déclarer un tel tableau en MIPS, le mot clé `.word` est utilisé afin d'allouer en mémoire un mot de 32 bits et de lui donner la valeur donnée en argument. L'exemple suivant montre la déclaration d'un tableau de 10 entiers. Le label `tab` permet de trouver l'adresse a de la première cellule du tableau. Ainsi, à l'adresse a se trouve la valeur 1, à l'adresse $a + 4$ se

Examen du cours de Fonctionnement des Ordinateurs
1ère Session, Juin 2017
Partie 1

NOM : ..... PRENOM : ..... SECTION : .....

trouve la valeur 7, à l'adresse $a + 8$ se trouve la valeur 19, etc.

```
1         .data
2 tab:
3         .word   1
4         .word   7
5         .word   19
6         .word   223
7         .word   511
8         .word   517
9         .word   603
10        .word   1020
11        .word   1021
12        .word   1022
```

Pour accéder à une cellule d'index i d'un tel tableau, il suffit de connaître l'adresse de la première cellule a et la taille d'une cellule t (ici, 4 octets). L'adresse de la cellule d'index i est égale à $a + i * t$. Le code suivant montre comment accéder à la cellule d'index $i = 5$ dans le tableau `tab` déclaré ci-dessus. L'instruction `sll` (*shift left logical*) est utilisée pour multiplier l'index de la cellule par la taille de la cellule ($t = 4$), en effectuant un décalage de 2 bits vers la gauche.

```
1         la      $a0, tab        # adresse de tab dans $a0
2         li      $a1, 5          # index de la cellule (5) dans $a1
3         sll     $t0, $a1, 2     # Calcule deplacement adresse (4*5)
4         add     $t0, $t0, $a0   # Ajoute adresse de base
5         lw      $t1, 0($t0)     # Charge le contenu de la cellule (517)
```

Ce qui vous est demandé : Il vous est demandé de fournir une implémentation en langage d'assemblage MIPS de la fonction récursive `recherche`. **Cette implémentation doit absolument être récursive. Une implémentation non récursive sera considérée comme incorrecte et ne rapportera aucun point !** Votre fonction doit respecter la convention d'appel suivante : l'adresse de la première cellule du tableau est passée dans le registre `a0`, l'index le plus bas du tableau est passé dans le registre `a1`, l'index le plus haut dans le registre `a2`, le nombre recherché dans le registre `a3` et la valeur de retour dans le registre `v0`. A titre d'exemple, le code suivant appelle la fonction `recherche` sur le tableau `tab` de 10 entiers déclaré ci-dessus. L'entier recherché est 603. La fonction `recherche` devrait retourner la valeur 6 dans le registre `v0`.

```
1         .text
2 main:   la      $a0, tab        # $a0 : adresse tableau
3         li      $a1, 0          # $a1 : index bas
4         li      $a2, tab_size   # $a2 : index haut (taille tableau-1)
5         addi    $a2, -1
6         li      $a3, 603        # $a3 : element recherche (ici 603)
7         ...
8         jal     recherche
9         ...                     # le resultat se trouve dans $v0
10
11        jr      $ra
12
13 recherche:
14   ...                  # a implementer
```

Examen du cours de Fonctionnement des Ordinateurs
1ère Session, Juin 2017
Partie 1

NOM : . . . . . PRENOM : . . . . . SECTION : . . . . . .

Q1 **(implémentation en langage d'assemblage MIPS de la fonction `recherche`)**

Examen du cours de Fonctionnement des Ordinateurs
1ère Session, Juin 2017
Partie 1

NOM : ..... PRENOM : ..... SECTION : .....

Q1 **(suite)**

Question 2 – IEEE754 (/4)

Cette question concerne l'addition de nombres flottants représentés en utilisant le standard IEEE 754. La question utilise les mêmes principes que ceux d'IEEE 754 mais avec des tailles de mantisse et d'exposant différentes. La taille M de la mantisse est 4 bits alors que la taille E de l'exposant est 3 bits. Le biais B vaut 3. Les arrondis sont effectués selon *round-to-nearest-even*.

Seule une représentation normalisée est supportée. La valeur spéciale où l'exposant vaut $e = 7$ et la mantisse vaut $m = 15$ indique un nombre non représentable.

Ce qui vous est demandé :

Q2a Soit le nombre $x = 13,5$. Donnez en binaire les valeurs de la mantisse et de l'exposant de l'approximation $\hat{x}$ de x. Donnez le détail du calcul.

Q2b Soit le nombre $y = 5,75$. Donnez en binaire les valeurs de la mantisse et de l'exposant de l'approximation $\hat{y}$ de y. Donnez le détail du calcul.

Q2c Effectuez l'addition en virgule flottante des représentations $\hat{x}$ et $\hat{y}$. Indiquez le détail de votre calcul. Donnez en binaire les valeurs de la mantisse et de l'exposant du résultat $\widehat{x + y}$, ainsi que sa valeur en décimal.

Q2d Calculez l'erreur absolue du résultat $\widehat{x + y}$.

Examen du cours de Fonctionnement des Ordinateurs
1ère Session, Juin 2017
Partie 1

NOM : ..... PRENOM : ..... SECTION : .....

Q2e Donnez la taille minimum de la mantisse M' nécessaire pour que l'erreur absolue soit nulle pour la représentation et l'addition des valeurs utilisées dans cet exercice ($x = 13,5$ et $y = 5,75$).

Q2a **(représentation de x)**

Q2b **(représentation de y)**

Q2c **(représentation de $\widehat{x+y}$)**

Q2d **(erreur absolue pour $\widehat{x+y}$)**

Q2e **(taille minimale de mantisse M')**

Examen du cours de Fonctionnement des Ordinateurs
1ère Session, Juin 2017
Partie 2

NOM : ..... PRENOM : ..... SECTION : .....

Consignes à lire impérativement !

L'examen est composé de **2 parties**. Chaque partie dure **2 heures**. Il vous est demandé de respecter les consignes suivantes.

- Commencez par écrire vos **nom**, **prénom** et **section** (math, info, ...) sur chaque feuille, y compris les feuilles de brouillon.
- Laissez vos calculatrice, téléphone portable et notes de cours dans votre sac. Leur usage n'est **pas autorisé**. Pensez à éteindre votre téléphone portable !
- Faites attention à la clarté et à l'organisation de vos réponses. Respectez les règles grammaticales et orthographiques.
- Utilisez pour vos réponses les **cadres** prévus à cet effet. Si davantage d'espace est nécessaire, utilisez le dos de la feuille ou une feuille supplémentaire et indiquez clairement où se situe le restant de la réponse.
- Vous devez terminer cette partie de l'examen avant de pouvoir sortir de la salle (pour aller à la toilette par exemple).
- Toutes les feuilles (énoncé et brouillon) doivent être remises en fin d'examen.
- Vérifiez que vous avez répondu à toutes les questions (il y a **3 questions** dans cette partie).

Question 1 – Cache *set-associative* (/4)

Soit un système informatique composé d'une mémoire RAM et d'une cache.

- La mémoire RAM a une taille de 65536 octets organisés en 65536 mots de 8 bits.
- La cache a une taille de 16 octets organisés en 4 lignes de 4 octets. La cache est de type *set-associative*. Son degré d'associativité est 2 (elle contient par conséquent 2 sets). La stratégie de remplacement de la cache est *least recently used* (LRU) : la ligne remplacée est celle dont le dernier usage est le plus éloigné dans le temps.

L'objectif de cette question est de déterminer le *hit ratio* de cette cache pour la séquence de lectures aux adresses suivantes : `0xA3C8`, `0xA3C9`, `0xA3CA`, `0xB5E0`, `0xB5E4`, `0xB5E8`, `0xB5EC`, `0xA3C8`. On considère que la cache est initialement vide.

Il vous est demandé de :

Q1a Indiquer la correspondance entre une adresse mémoire (16 bits) et un emplacement en cache, en identifiant les bits de l'adresse qui correspondent au *tag*, au *set* et à l'*offset*. A cet effet, remplissez chaque case (bit) de l'adresse par la lettre T (*tag*) , S (*set*) ou O (*offset*).

Q1b Exécuter manuellement la séquence de lectures. Pour chaque lecture

- donnez les valeurs de *tag*, *set* et *offset*, en hexadécimal.

Examen du cours de Fonctionnement des Ordinateurs
1ère Session, Juin 2017
Partie 2

NOM : ..... PRENOM : ..... SECTION : .....

- indiquez s'il s'agit d'un *hit* ou d'un *miss*. Si une ligne est remplacée, indiquez laquelle (identifiez la avec *set* et *tag* actuel).

Q1c Calculer le *hit ratio* de la séquence de lectures.

Q1d Calculer le temps maximum nécessaire pour effectuer une lecture en mémoire cache pour que le système RAM/-cache soit efficace. On vous indique à cet effet que la lecture en RAM d'une ligne de cache (4 octets) nécessite 80ns. Indiquez votre raisonnement ! ! !

Q1a **(structure d'une adresse)**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

Q1b **(rejeu de la trace d'accès)**

Adresse	Tag	Set	Offset	Hit / Miss
`0xA3C8`				
`0xA3C9`				
`0xA3CA`				
`0xB5E0`				
`0xB5E4`				
`0xB5E8`				
`0xB5EC`				
`0xA3C8`				

Q1c **(*hit ratio*)**

.....

.....

Q1d **(temps maximum de lecture en cache)**

.....

.....

.....

Examen du cours de Fonctionnement des Ordinateurs
1ère Session, Juin 2017
Partie 2

NOM : ..... PRENOM : ..... SECTION : .....

Question 2 – Pilotage GPIO en langage d'assemblage MIPS (/4)

L'objectif de cette question est l'implémentation en langage d'assemblage d'un programme simulant une fonction logique XOR (ou-exclusif). Ce programme doit fonctionner sur un système embarqué articulé autour d'un microcontrôleur intégrant un coeur MIPS. Ce microcontrôleur est connecté à 3 LEDs nommées `LD1` à `LD3`. Il est également connecté à 2 interrupteurs nommés `SW1` à `SW2`.

Les interrupteurs `SW1` et `SW2` définissent les entrées de la fonction logique. Les LEDs `LD1` et `LD2` affichent les valeurs des entrées, i.e. `LD1`=`SW1` et `LD2`=`SW2`. La LED `LD3` affiche la sortie de la fonction, i.e. `LD3`= (`SW1` XOR `SW2`). Ainsi, si par exemple les valeurs des interrupteurs `SW1` à `SW2` valent respectivement `1`, `0`, alors les LEDs `LD1` à `LD3` valent respectivement `1`, `0`, `1`.

Afin d'interragir avec les interrupteurs et les LEDs, le microcontrôleur dispose d'un port d'entrées/sorties appelé `PORT1`. Ce port est capable de contrôler 32 broches du microcontrôleur. Seules 5 broches sur 32 sont utilisées pour lire/contrôler les interrupteurs et LEDs. L'attribution des broches aux LEDs et interrupteurs est reprise dans la table suivante. La configuration et l'état des autres broches doivent demeurer inchangés car ceux-ci sont reliées à d'autres composants.

Composant	Numéro de broche
`LD1`	0
`LD2`	1
`LD3`	2
`SW1`	8
`SW2`	9

Le port `PORT1` est contrôlé à l'aide de 3 registres différents nommés `DIR1`, `WRITE1` et `READ1`. Chacun de ces registres est accessible via le bus mémoire. Chaque registre comporte 32 bits. Pour chaque registre, le bit n concerne la broche n. La description, l'adresse et la valeur initiale de chaque registre sont fournies dans la table ci-dessous.

Registre	Adresse	Description	Valeur initiale
`DIR1`	`0xF0000000`	Détermine la direction des broches du port. Si le bit n vaut 1, cela signifie que la broche n est une sortie. Dans le cas contraire, cette broche est une entrée.	0
`WRITE1`	`0xF0000004`	Permet de changer la valeur d'une broche configurée en sortie. Le bit n détermine l'état de la broche n.	0
`READ1`	`0xF0000008`	Permet de lire l'état d'une broche configurée en entrée. Le bit n contient l'état de la broche n.	—

Examen du cours de Fonctionnement des Ordinateurs
1ère Session, Juin 2017
Partie 2

NOM : ..... PRENOM : ..... SECTION : .....

Ce qui vous est demandé

Premièrement, fournissez des suites d'instructions MIPS permettant respectivement d'allumer et éteindre la LED `LD2`. Ces extraits ne doivent pas s'occuper de configurer en sortie la broche connectée à la LED `LD2`. L'état des autres broches doit rester inchangé.

Deuxièmement implémentez en langage d'assemblage MIPS un programme simulant la fonction logique XOR de la façon décrite ci-dessus. Votre programme commence au label `main` désignant la première instruction à exécuter, puis s'exécute en boucle, sans fin. Le programme est responsable de la configuration des broches en entrée ou en sortie.

Q2a Code MIPS allumant `LD2`

Q2b Code MIPS éteignant `LD2`

Q2c Programme MIPS simulant la fonction logique XOR.

Q2a **(instructions MIPS pour allumer la LED `LD2`)**

Q2b **(instructions MIPS pour éteindre la LED `LD2`)**

Examen du cours de Fonctionnement des Ordinateurs
1ère Session, Juin 2017
Partie 2

NOM : . . . . . PRENOM : . . . . . SECTION : . . . . . .

Q2c **(programme simulant XOR en langage d'assemblage MIPS)**

Examen du cours de Fonctionnement des Ordinateurs
1ère Session, Juin 2017
Partie 2

NOM : ..... PRENOM : ..... SECTION : .....

Question 3 – Types de données et représentation mémoire (/2)

Cette question concerne la représentation de données multi-octets en mémoire. Soit une mémoire de 4 Ko organisée sous la forme de 4096 cellules de 8 bits. Une partie du contenu de cette mémoire est donnée dans la Table 1. Les adresses et valeurs des mots sont représentées en hexadécimal. La mémoire est accédée par un processeur qui ne peut lire ou écrire que 8 bits à la fois. Le processeur est de type *big-endian*.

Adresse mémoire	Donnée
...	...
0x120	0xAF
0x121	0x64
0x122	0xC8
0x123	0x2D
...	...

TABLE 1 – Contenu de la mémoire.

Le processeur exécute un programme qui effectue des lectures d'entiers dans cette mémoire. Les entiers lus peuvent avoir des tailles différentes et peuvent être signés ou non-signés. Les nombres signés sont représentés en complément à 2. On distingue les types suivants : entier signé de 8 bits (`char`), entier non-signé de 8 bits (`unsigned char`), entier signé de 16 bits (`short`) et entier non-signé de 16 bits (`unsigned short`).

Ce qui vous est demandé : Le programme effectue des lectures d'entiers de différents types situés à différentes adresses. Pour chaque lecture, donnez en décimal la valeur de l'entier lu.

Q3a **(`char` à l'adresse `0x120`)**

.....

Q3b **(`unsigned char` à l'adresse `0x121`)**

.....

Q3c **(`unsigned short` à l'adresse `0x122`)**

.....

Q3d **(`short` à l'adresse `0x122`)**

.....