Université d'Alger 1

Faculté des Sciences
Département Mathématiques et Informatique

Examen L2-ADO
12-Février-2019

Nom : Prénom : Matricule :		/ 40

Exercice 1 : (13 points)

Supposons que nous avons une machine avec des adresses mémoire sur 16 bits. Supposons aussi que nous avons un cache mémoire avec la configuration suivante :

- Cache 1-Associatif
- Taille du cache : 4 Ko
- Taille d'une ligne de cache : 2 mots (8 octets)

1/ Donner les largeurs en bits des champs Etiquette, Index et Offset (Mot et Octet) d'un bloc de cache

Etiquette : **4** (0,5 point) Index : **9** (0,5 point) Offset : **3** (0,5 point)

2/ Donnez les valeurs des champs Etiquette, Index et Offset pour les adresses mémoires suivantes (donner vos réponses en binaire en tenant compte des largeurs en bits proposées dans la question précédente) (9 points):

Adresse	Etiquette (1,5 points)	Index (1,5 points)	Offset (1,5 points)
0x1004	**0001**	**0000 0000 0**	**100**
0x0008	**0000**	**0000 0000 1**	**000**
0x100C	**0001**	**0000 0000 1**	**100**

3/ Supposons que le cache soit initialement vide et que nous accédions aux adresses suivantes, dans cette ordre :

0x0000, 0x0004, 0x0008, 0x000C, 0x1000, 0x1004, 0x1008, 0x100C, 0x0000, 0x0004, 0x1008, 0x100C

Identifier pour les adresses ci-dessus, les succès de cache et les échecs. Pour un échec, indiquer s'il s'agit de conflit (remplacement d'une ligne de cache) ou obligatoire (ligne de cache initialement vide).

adresse	Succès / Echec	Type d'Echec (Initial ou Remplacement)	
0x0000	**Echec**	**Obligatoire**	(0,5 point)
0x0004	**Succès**		(0,5 point)
0x0008	**Echec**	**Obligatoire**	(0,5 point)
0x000C	**Succès**		(0,5 point)
0x1000	**Echec**	**Conflit**	(0,5 point)
0x1004	**Succès**		(0,5 point)
0x1008	**Echec**	**Conflit**	(0,5 point)
0x100C	**Succès**		(0,5 point)
0x0000	**Echec**	**Conflit**	(0,5 point)
0x0004	**Succès**		(0,5 point)
0x1008	**Succès**		(0,5 point)
0x100C	**Succès**		(0,5 point)

4/ Si nous considérons un temps d'accès au cache de l'ordre de 2 cycles d'horloge pour les données trouvées en cache (succès) et une pénalité de l'ordre de 100 cycles d'horloge pour les données non trouvées en cache et qui requière donc un accès RAM.

- Donner le temps d'accès total aux adresses indiquées ci-dessus : **514 cycles** (0,5 point)
- Quel serait le temps d'accès total avec un taux de succès de 75% ? **318 cycles** (0,5 point)

Exercice 2 : (19 points)

Dans une équipe de reverse-engineering, votre patron vous a attribué la tache de désassembler le code d'un concurrent industriel et ce dans le but d'implémenter une solution similaire.

1/ Complétez le décodage des instructions machines indiquées dans les commentaires

2/ Compléter l'organigramme de la solution proposée

3/ Que fait donc ce programme ? (4 points)

Votre réponse ici ne sera prise en compte que si votre organigramme est correct.

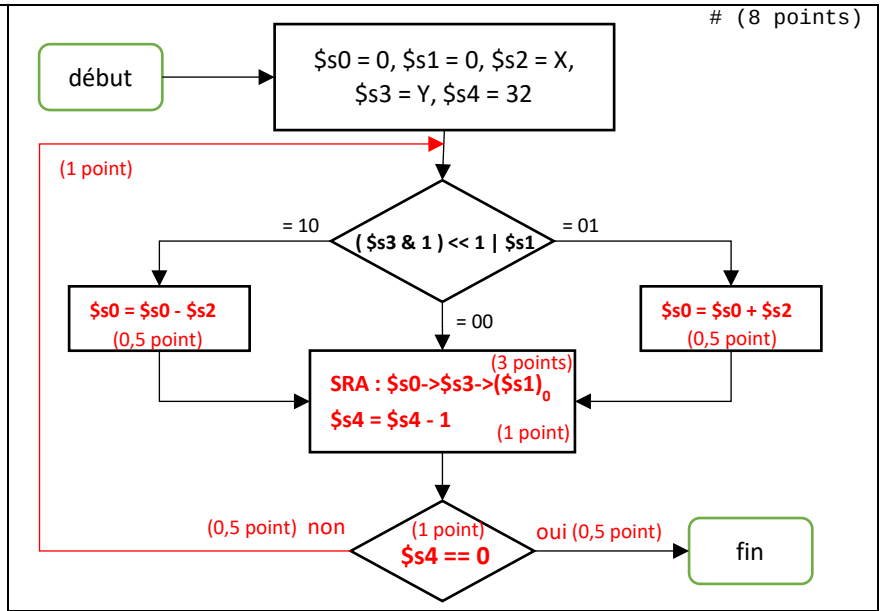
Multiplication **(1point)** de nombres entiers signés **(2 points)** sur 32 bits **(1 point)**

```
   Adresse de
  l'instruction
                    .data
                       X:   .word -153
                       Y:   .word  6

                    .text
    0x0040 0000         move $s0, $0
    0x0040 0004         move $s1, $0
    0x0040 0008         la   $s2, X
    0x0040 000C
    0x0040 0010         lw   $s2, ($s2)
    0x0040 0014         la   $s3, Y
    0x0040 0018
    0x0040 001C         lw   $s3, ($s3)
    0x0040 0020         addi $s4, $0, 32

    0x0040 0024   DO:   andi $t0, $s3, 1
    0x0040 0028         sll  $t0, $t0, 1
    0x0040 002C         or   $t0, $t0, $s1
    0x0040 0030         addi $t0, $t0, -1
    0x0040 0034         bne  $t0, $0, FI
    0x0040 0038         add  $s0, $s0, $s2
    0x0040 003C         j OK
    0x0040 0040   FI:   addi $t0, $t0, -1
    0x0040 0044         bne  $t0, $0, OK
    0x0040 0048         sub  $s0, $s0, $s2

    0x0040 004C   OK:   andi $t0, $s0, 1       # Instructions machines à décoder
    0x0040 0050         andi $s1, $s3, 1       # Ajouter des étiquettes au code si nécessaire !
    0x0040 0054         srl  $s3, $s3, 1       # <-- 0x00139842      (1,5 point)
    0x0040 0058         sll  $t0, $t0, 31      # <-- 0x000847C0      (1,5 point)
    0x0040 005C         or   $s3, $s3, $t0     # <-- 0x02689825      (1,5 point)
    0x0040 0060         sra  $s0, $s0, 1       # <-- 0x00108043      (1,5 point)
    0x0040 0064         addi $s4, $s4,-1       # <-- 0x2294ffff      (1,5 point)
    0x0040 0068         bne  $s4, $0, DO       # <-- 0x1414ffee      (1,5 point)
```



Exercice 3 : (8 points)

Proposer un fragment de code en assembleur MIPS pour compter le nombre d'occurrences de caractères alphabétiques (minuscules ou majuscules) dans une chaîne de caractères. Supposer que l'adresse de base de la chaîne est dans le registre **$s0** et le nombre retourné doit être dans **$s1**. Supposer aussi que la chaine de caractères se termine par le caractère nul (zéro). **(4 points pour un programme fonctionnel, 4 points pour la conversion)**

```
      ori $t2,$zero, 0x41     # le code Ascii du caractère A dans t2
      ori $t3,$zero, 0x5A     # le code Ascii du caractère Z dans t3
      xor $s1, $s1, $s1       # initialisation du compteur à zéro
      move $t0, $s0           # $t0 pointe vers le premier caractère de
Loop:                         # la chaine
      lb $t1, ($t0)           # lire un octet dans $t1
      beq $t1, $zero, Exit    # sortir si le caractère nul est détecté
      andi $t1, $t1, 0xDF     # Conversion éventuelle en majuscule
      blt $t1, $t2, Skip      # ignorer si inférieur à 0x41 (car. non alphabétique)
      bgt $t1, $t3, Skip      # ignorer si supérieur à 0x5A (car. non alphabétique)
      addiu $s1, $s1, 1       # incrémenter le compteur
Skip:
      addi $t0, $t0, 1        # avancer le pointeur sur la chaine de caractère
      j Loop
Exit:
```

ASCII TABLE

Decimal	Hex	Char	Decimal	Hex	Char	Decimal	Hex	Char	Decimal	Hex	Char
0	0	[NULL]	32	20	[SPACE]	64	40	@	96	60	`
1	1	[START OF HEADING]	33	21	!	65	41	A	97	61	a
2	2	[START OF TEXT]	34	22	"	66	42	B	98	62	b
3	3	[END OF TEXT]	35	23	#	67	43	C	99	63	c
4	4	[END OF TRANSMISSION]	36	24	$	68	44	D	100	64	d
5	5	[ENQUIRY]	37	25	%	69	45	E	101	65	e
6	6	[ACKNOWLEDGE]	38	26	&	70	46	F	102	66	f
7	7	[BELL]	39	27	'	71	47	G	103	67	g
8	8	[BACKSPACE]	40	28	(	72	48	H	104	68	h
9	9	[HORIZONTAL TAB]	41	29	)	73	49	I	105	69	i
10	A	[LINE FEED]	42	2A	*	74	4A	J	106	6A	j
11	B	[VERTICAL TAB]	43	2B	+	75	4B	K	107	6B	k
12	C	[FORM FEED]	44	2C	,	76	4C	L	108	6C	l
13	D	[CARRIAGE RETURN]	45	2D	-	77	4D	M	109	6D	m
14	E	[SHIFT OUT]	46	2E	.	78	4E	N	110	6E	n
15	F	[SHIFT IN]	47	2F	/	79	4F	O	111	6F	o
16	10	[DATA LINK ESCAPE]	48	30	0	80	50	P	112	70	p
17	11	[DEVICE CONTROL 1]	49	31	1	81	51	Q	113	71	q
18	12	[DEVICE CONTROL 2]	50	32	2	82	52	R	114	72	r
19	13	[DEVICE CONTROL 3]	51	33	3	83	53	S	115	73	s
20	14	[DEVICE CONTROL 4]	52	34	4	84	54	T	116	74	t
21	15	[NEGATIVE ACKNOWLEDGE]	53	35	5	85	55	U	117	75	u
22	16	[SYNCHRONOUS IDLE]	54	36	6	86	56	V	118	76	v
23	17	[ENG OF TRANS. BLOCK]	55	37	7	87	57	W	119	77	w
24	18	[CANCEL]	56	38	8	88	58	X	120	78	x
25	19	[END OF MEDIUM]	57	39	9	89	59	Y	121	79	y
26	1A	[SUBSTITUTE]	58	3A	:	90	5A	Z	122	7A	z
27	1B	[ESCAPE]	59	3B	;	91	5B	[	123	7B	{
28	1C	[FILE SEPARATOR]	60	3C	<	92	5C	\	124	7C	|
29	1D	[GROUP SEPARATOR]	61	3D	=	93	5D	]	125	7D	}
30	1E	[RECORD SEPARATOR]	62	3E	>	94	5E	^	126	7E	~
31	1F	[UNIT SEPARATOR]	63	3F	?	95	5F	_	127	7F	[DEL]