TP 2 : Architecture des ordinateurs

Prise en main du simulateur MARS, Comprendre l'organisation et l'exécution du code d'assemblage MIPS, Se familiariser l'architecture externe du MIPS via un environnement virtuel.

Durée : 2 semaines

Les programmes C devront respecter les conventions d'écriture d'un programme C (commentaires, indentation, etc.), être compilés à partir de Dev c++.

Exercice 1

Créer un nouveau fichier avec l'éditeur : **File → New**
Saisir le code d'assemblage suivant :

Edit | Execute

mips1.asm* | mips2.asm | Downloads

```
1  ######################## Segment data ###################################
2  .data
3  theString:
4      .space 64                    # reservation de 64 octets dans la mémoire pour la chaine theString
5
6  ######################## Segment text ###################################
7      .text
8  main:
9      li      $v0, 8               # chargement immediat de la valeur 8 dans $v0
10                                  # pour la lecture d'une chaine de caractères
11     la      $a0, theString       # chargement de l'adresse de la chaine de caractères dans $a0
12     li      $a1, 64              # $a1 contient la longeur de la chaine de caractères
13     syscall
14 ###################### Fin du programme ################################
15     li $v0, 10
16     syscall
```

1. Faire l'assemblage du programme : **Run → Assemble** puis exécuter le programme par le bouton **« Run the current program ».**
2. Dans la fenêtre **« Run I/O »** en bas de l'écran saisir la chaine de caractères « hello ».
3. Vérifier le contenu du registre $v0. Traduire la valeur obtenue en décimal.
4. Vérifier le contenu du registre $a1. Traduire la valeur obtenue en décimal.
5. Vérifier le contenu du registre $a0. Que signifie la valeur obtenue ?
6. Dans la fenêtre **« Data segment »** lire les deux colonnes Value(+0) et Value(+4) qui correspondent à la ligne d'adresse 0x10010000.
7. Traduire ces deux valeurs en code ASCII. Quel est le résultat obtenu ? utiliser la table ASCII (voir l'annexe).

Exercice 2

Etant donné le code en C suivant :

```
#include <stdio.h>
int tableau[40];
int main() {
  int t0, t1, t2, t3, t4, t5, t6, t7;  /* registres temporaires */
  t6 = 1;   t7 = 1;
  tableau[0] = t6;  /* Rangement du deux premiers éléments de  */
```

```
  tableau[t7] = t6; /* la séquence dans le tableau          */
  t0 = 2;
boucle:
  t3 = t0 - 2;   t4 = t0 - 1;
  t1 = tableau[t3];   t2 = tableau[t4];
  t5 = t1 + t2;
  tableau[t0] = t5;
  t0 = t0 + 1;
  if (t0 < 40) goto boucle ;
  return 0; }
```

Ecrire le code d'assemblage correspondant.

- Notons que chaque élément du tableau est stocké dans mot (4 octets). Réserver un espace mémoire au notre tableau.
- Utiliser l'opcode « li » pour initialiser t6 et t7.
- Utiliser l'opcode « sw » pour le rangement dans la mémoire (initialisation des deux premiers termes du tableau).
- Utiliser l'opcode « lw » pour le rangement dans t1 et t2.
- Commenter votre code.

Annexe

Hex	Caractère	
0	NUL	
1	SOH	(Null char.)
2	STX	(Start of Header)
3	ETX	(Start of Text)
4	EOT	(End of Text)
5	ENQ	(End of Transmission)
6	ACK	(Enquiry)
7	BEL	(Acknowledgment)
8	BS	(Bell)
9	HT	(Backspace)
0A	LF	(Horizontal Tab)
0B	VT	(Line Feed)
0C	FF	(Vertical Tab)
0D	CR	(Form Feed)
0E	SO	(Carriage Return)
0F	SI	(Shift Out)
10	DLE	(Shift In)
11	DC1	(Data Link Escape)
12	DC2	(XON)(Device Control 1)
13	DC3	(Device Control 2)
14	DC4	(XOFF)(Device Control 3)
15	NAK	(Device Control 4)
16	SYN	(Negative Acknowledgement)
17	ETB	(Synchronous Idle)
18	CAN	(End of Trans. Block)
19	EM	(Cancel)
1A	SUB	(End of Medium)
1B	ESC	(Substitute)
1C	FS	(Escape)
1D	GS	(File Separator)
1E	RS	(Group Separator)
1F	US	(Request to Send)(Record Separator)
20	SP	(Unit Separator)
21	!	(Space)
22	"	(exclamation mark)
23	#	(double quote)
24	$	(number sign)
25	%	(dollar sign)
26	&	(percent)
27	'	(ampersand)
28	(	(single quote)
29	)	(left opening parenthesis)
2A	*	(right closing parenthesis)
2B	+	(asterisk)
2C	,	(plus)
2D	-	(comma)
2E	.	(minus or dash)
2F	/	(dot)
30	0	(forward slash)

Hex	Caractère	
31	1	
32	2	
33	3	
34	4	
35	5	
36	6	
37	7	
38	8	
39	9	
3A	:	
3B	;	(colon)
3C	<	(semi-colon)
3D	=	(less than sign)
3E	>	(equal sign)
3F	?	(greater than sign)
40	@	(question mark)
41	A	(AT symbol)
42	B	
43	C	
44	D	
45	E	
46	F	
47	G	
48	H	
49	I	
4A	J	
4B	K	
4C	L	
4D	M	
4E	N	
4F	O	
50	P	
51	Q	
52	R	
53	S	
54	T	
55	U	
56	V	
57	W	

Hex	Caractère	
58	X	
59	Y	
5A	Z	
5B	[	
5C	\	(left opening
5D	]	(back slash)
5E	^	(right closing
5F	_	(caret cirumf
60	`	(underscore)
61	a	
62	b	
63	c	
64	d	
65	e	
66	f	
67	g	
68	h	
69	i	
6A	j	
6B	k	
6C	l	
6D	m	
6E	n	
6F	o	
70	p	
71	q	
72	r	
73	s	
74	t	
75	u	
76	v	
77	w	
78	x	
79	y	
7A	z	
7B	{	
7C	|	(left opening
7D	}	(vertical bar)
7E	~	(right closing
7F	DEL	(tilde)