FONAMENTS DE COMPUTADORS
Pràctica 7
Assemblador: dades en memòria, representació i accés

INTRODUCCIÓ I OBJECTIUS

En la present pràctica continuem amb la programació en assemblador fent ús del simulador PCSpim. Els objectius són:

- Desenvolupar programes amb declaració de dades em memòria i que utilitzen dades emmagatzemades en memòria
- Comprendre la representació de diferents tipus de dades: enters, reals i caràcters
- Introduir l'ús de pseudoinstruccions
- Utilitzar instruccions per a l'accés a memòria

NOTA: Perquè el PCSPIM carregue correctament les pseudoinstruccions, s'ha de configurar (per mitjà del menú *Simulator→ Settings*) de la següent forma: Desactivar l'opció *Bare Machine* i activar l'opció *Allow pseudo instructions*.

ÚS DE LES DIRECTIVES DE DADES

Les directives de dades són indicacions al programa assemblador (el programa que tradueix des de llenguatge assemblador a codi màquina), que li seran d'utilitat en el moment de generar codi executable. Aquestes directives no generen codi de màquina, és a dir, no generen instruccions. El significat d'algunes de les directives més utilitzades es mostra en la taula següent:

Directiva	Significat
.data <addr>	Emmagatzema els elements declarats en el segment de dades, a partir de l'adreça indicada en addr.
.byte b1,..,bn	Emmagatzema n valors enters en paraules successives de 8 bits.
.half h1,..,hn	Emmagatzema n valors enters en paraules successives de 16 bits.
.word w1,..,wn	Emmagatzema n valors enters en paraules successives de 32 bits.
.float f1,..,fn	Emmagatzema n valors reals en paraules successives de 32 bits.
.ascii str	Emmagatzema la cadena str en memòria. Un byte per caràcter.
.asciiz str	Emmagatzema la cadena str en memòria. Un byte per caràcter i afegeix al final el caràcter NULL.

Els enters es representen, tots, en complement a dos. Els reals, en format IEEE754 de simple precisió. Els caràcters s'emmagatzemen en ASCII de 8 bits

Quan carreguem un programa que inclou directives de dades, l'efecte d'aquestes directives es veu immediatament en el segment de dades del simulador. Apareixerà la reserva i inicialització de posiciones de memòria.

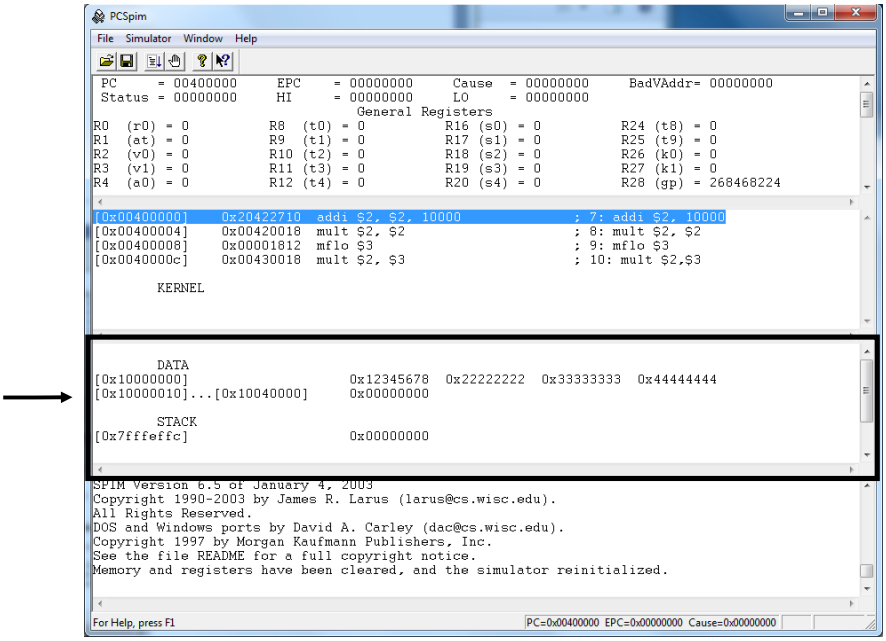

Figura 1. Localització de la finestra de segment de dades dins l'entorn gràfic del simulador PCSpim.

Finestra de segment de dades: en aquesta finestra es mostra el contingut de la memòria. La memòria es desplega i mostra rangs d'adreces i el contingut d'aquest rang. De manera que una línia amb la informació següent:

[0x10000000] 0x12345678 0x22222222 0x3333333 0x44444444

S'interpreta de la forma següent:

- Es mostra el contingut de les quatre paraules de memòria que estan en les adreces 0x10000000 a 0x1000000C. La grandària d'una paraula és de 32 bits. Per tant, en una línia es mostren 32x4 = 128 bits.
- La paraula de memòria 0x10000000 té el contingut 0x12345678 (32 bits)
  - Els bytes que componen aquesta paraula ocuparan, per tant, les posicions següents segons el format *little endian*:

    [0x10000000] 0x78
    [0x10000001] 0x56
    [0x10000002] 0x34
    [0x10000003] 0x12
- La paraula de memòria 0x10000004 té el contingut 0x22222222 (32 bits)
- La paraula de memòria 0x10000008 té el contingut 0x33333333 (32 bits)
- La paraula de memòria 0x1000000C té el contingut 0x44444444 (32 bits)

ALINEACIÓ DE LES DADES EN MEMÒRIA

L'alineació de dades en memòria és un requisit de molts processadors. Aquesta alineació de les dades vol dir que quan el processador accedeix a una dada d'un determinat tipus (byte, half o word), ho farà emprant direccions amb unes característiques concretes i particulars. Per exemple, quan el processador vol llegir una dada de tipus word, l'adreça cal que siga múltiple de quatre, ja que els words ocupen quatre bytes. En canvi, les adreces que fan referència a dades de tipus hal cal que siguen múltiples de dos, mentre que les de tipus byte no tenen cap restricció.
Per a mantindre la relació entre el tipus de dada i la seva adreça, el programa assemblador, al carregar el programa i processar les directives, deixarà els espais necessaris entre dades declarades consecutivament.

Considereu el codi següent i responngueu a les qüestions següents:

```
.data 0x10000000
.byte 1,-1,2
.half -2, 3
.byte 4
.word  -4
.byte 5
.half 6
```

Pregunta 1. Ompliu **de forma teòrica** la taula següent amb els continguts que tindrà la memòria al carregar i processar les directives.

	Contingut (Decimal i hexadecimal)			
Adreça (HEX)	31 ⋯ 24	23 ⋯ 16	15 ⋯ 8	7 ⋯ 0
0x10000000	00000000	0000010	10000001	00000001
0x10000004	00000000	00000011	10000000	00000010
0x10000008	00000000	00000000	0000000	00000100
0x1000000C	10000000	00000000	00000000	00000100
0x10000010	00000000	00000110	00000000	00000101
0x10000014				

Pregunta 2. **Una vegada al laboratori**, teclegeu i carregueu el codi anterior en el PCSpim, i ompliu novament la taula, aquesta vegada copiant les dades tal com apareixen en la finestra del segment de dades.

	Contingut (Decimal i hexadecimal)			
Adreça (HEX)	31 ⋯ 24	23 ⋯ 16	15 ⋯ 8	7 ⋯ 0
0x10000000				
0x10000004				
0x10000008				
0x1000000C				
0x10000010				
0x10000014				

REPRESENTACIÓ DE REALS

Els nombres reals es representen en la majoria de computadors amb el format IEEE754 de simple precisió. En aquest format, s'utilitzen els criteris següents per representar un nombre real:

- El format és:

1 bit	8 bits	23 bits
Signe	Exponent	Mantissa

- El bit de signe a zero vol dir que la quantitat representada és positiva. I amb un u, vol dir que és negativa.
- L'exponent es representa en excés Z, amb Z=127, Aço vol dir que si l'exponent de la quantitat que volem representar es 3, el que realment s'emmagatzemarà és 130.
- La mantissa cal que estiga normalitzada de la forma 1.xxx, i s'emmagatzema emprant la tècnica del bit implícit. Es a dir, el primer 1 no s'emmagatzema.

Teclegeu el codi següent en un editor, carregeu-lo en el PCSpim, i responegueu les qüestions següents:

```
.data 0x10000000
.float  1.0 , -1.0 , 3.23, -3.23
```

Pregunta 3. Ompliu la taula següent amb els continguts de la memòria mostrada en la finestra del segment de dades.

	Contingut (Decimal i hexadecimal)			
Adreça (HEX)	31 ⋯ 24	23 ⋯ 16	15 ⋯ 8	7 ⋯ 0
0x10000000				
0x10000004				
0x10000008				
0x1000000C				
0x10000010				

Pregunta 4. Indiqueu els camps de signe, exponent i mantissa (en binari tots tres) del quatre valors reals definits en el codi anterior:

	Signe	Exponent	Mantissa
1,00			
-1,00			
3,23			
-3.23			

REPRESENTACIÓ DE CARÀCTERS

Teclegeu el codi següent en un editor, carregeu-lo en el PCSpim, i respongueu les qüestions següents:

```
.data 0x10000000
.asciiz “el profe”
.byte 16
.ascii “el profe”
.byte 16
```

Pregunta 5. Ompliu la taula següent amb els continguts de la memòria mostrada en la finestra del segment de dades.

	Contingut (Decimal i hexadecimal)			
Adreça (HEX)	31 ⋯ 24	23 ⋯ 16	15 ⋯ 8	7 ⋯ 0
0x10000000				
0x10000004				
0x10000008				
0x1000000C				
0x10000010				

Pregunta 6. En quina adreça de memòria es troben les lletres “p”.

Adreça en hexadecimal:

LOCALITZACIÓ D'INSTRUCCIONS I DE DADES

El codi que es va a emprar en les següents preguntes és el següent. Escrigue-ho amb un editor de textos, anomeneu a l'arxiu "practica_7_1.s, carregueu-ho en el PCSPIM i comproveu que es carrega correctament.

```
.globl __start
.data 0x10000000
base:   .word 3
altura: .word 10
area:   .space 4

.text 0x00400000
__start:
      la $10, base
      lw $11, 0($10)
      la $12, altura
      lw $13, 0($12)
      mult $11,$13
      mflo $14
      addi $15,$0,2
      div $14,$15
      mflo $16
      la $17, area
      sw $16, 0($17)
.end
```

Codi 1. Càlcul de l'àrea d'un triangle (arxiu "practica_7_1.s")

Pregunta 7. Determineu les adreces de memòria associades amb les etiquetes següents:

Dada	Adreça
`base`	0x10000000
`altura`	
`area`	

Pregunta 8. Determineu les adreces de memòria on s'emmagatzemen les instruccions següents:

Instrucció	Adreça
`lui $10, 0x1000`	0x00400000
`lw $11, 0($10)`	
`lui $1, 0x1000`	
`ori $12, $1, 0x0004`	
`lw $13, 0($12)`	
`div $14,$15`	

Pregunta 9. Indiqueu el contingut del segment de dades abans d'iniciar-se l'execució, tenint en compte que les dades s'emmagatzemen en format "little endian". El contingut ha de posar-se en hexadecimal per cada byte de memòria.

31 … 24	23 … 16	15 … 8	7 … 0	Adreça

Pregunta 10. Indiqueu el contingut del segment de dades una vegada finalitzada l'execució, tenint en compte que les dades s'emmagatzemen en format "little endian". El contingut ha de posar-se en hexadecimal per cada byte de memòria.

31 … 24	23 … 16	15 … 8	7 … 0	Dirección

Pregunta 11. Determineu el contingut dels següents registres quan s'haja finalitzat l'execució del programa.

Registre	**Contingut**	**Què representa el contingut**
$10		
$11		
$12		
$13		
$14		
$15		
$16		
$17		

Pregunta 12. El programa proposat utilitza molts registres. No obstant això, és possible realitzar els mateixos càlculs amb menys registres. Reescriviu el programa en un arxiu anomenat "practica_7_2.s" on es minimitze el nombre de registres emprats.

Regular ASCII Chart (character codes 0 – 127)

000d	00h		(nul)	016d	10h	►	(dle)	032d	20h	␣	048d	30h	0	064d	40h	@	080d	50h	P	096d	60h	‘	112d	70h	p
001d	01h	☺	(soh)	017d	11h	◄	(dc1)	033d	21h	!	049d	31h	1	065d	41h	A	081d	51h	Q	097d	61h	a	113d	71h	q
002d	02h	☻	(stx)	018d	12h	↕	(dc2)	034d	22h	"	050d	32h	2	066d	42h	B	082d	52h	R	098d	62h	b	114d	72h	r
003d	03h	♥	(etx)	019d	13h	‼	(dc3)	035d	23h	#	051d	33h	3	067d	43h	C	083d	53h	S	099d	63h	c	115d	73h	s
004d	04h	♦	(eot)	020d	14h	¶	(dc4)	036d	24h	$	052d	34h	4	068d	44h	D	084d	54h	T	100d	64h	d	116d	74h	t
005d	05h	♣	(enq)	021d	15h	§	(nak)	037d	25h	%	053d	35h	5	069d	45h	E	085d	55h	U	101d	65h	e	117d	75h	u
006d	06h	♠	(ack)	022d	16h	▬	(syn)	038d	26h	&	054d	36h	6	070d	46h	F	086d	56h	V	102d	66h	f	118d	76h	v
007d	07h	•	(bel)	023d	17h	↨	(etb)	039d	27h	'	055d	37h	7	071d	47h	G	087d	57h	W	103d	67h	g	119d	77h	w
008d	08h	◘	(bs)	024d	18h	↑	(can)	040d	28h	(	056d	38h	8	072d	48h	H	088d	58h	X	104d	68h	h	120d	78h	x
009d	09h		(tab)	025d	19h	↓	(em)	041d	29h	)	057d	39h	9	073d	49h	I	089d	59h	Y	105d	69h	i	121d	79h	y
010d	0Ah	◙	(lf)	026d	1Ah		(eof)	042d	2Ah	*	058d	3Ah	:	074d	4Ah	J	090d	5Ah	Z	106d	6Ah	j	122d	7Ah	z
011d	0Bh	♂	(vt)	027d	1Bh	←	(esc)	043d	2Bh	+	059d	3Bh	;	075d	4Bh	K	091d	5Bh	[	107d	6Bh	k	123d	7Bh	{
012d	0Ch		(np)	028d	1Ch	└	(fs)	044d	2Ch	’	060d	3Ch	<	076d	4Ch	L	092d	5Ch	\	108d	6Ch	l	124d	7Ch	|
013d	0Dh	♪	(cr)	029d	1Dh	↔	(gs)	045d	2Dh	-	061d	3Dh	=	077d	4Dh	M	093d	5Dh	]	109d	6Dh	m	125d	7Dh	}
014d	0Eh	♫	(so)	030d	1Eh	▲	(rs)	046d	2Eh	.	062d	3Eh	>	078d	4Eh	N	094d	5Eh	^	110d	6Eh	n	126d	7Eh	~
015d	0Fh	☼	(si)	031d	1Fh	▼	(us)	047d	2Fh	/	063d	3Fh	?	079d	4Fh	O	095d	5Fh	_	111d	6Fh	o	127d	7Fh	⌂

Extended ASCII Chart (character codes 128 – 255) Latin1/CP1252

128d	80h	€		144d	90h			160d	A0h		176d	B0h	°	192d	C0h	À	208d	D0h	Ð	224d	E0h	à	240d	F0h	ð
129d	81h			145d	91h	‘		161d	A1h	¡	177d	B1h	±	193d	C1h	Á	209d	D1h	Ñ	225d	E1h	á	241d	F1h	ñ
130d	82h	‚		146d	92h	’		162d	A2h	¢	178d	B2h	²	194d	C2h	Â	210d	D2h	Ò	226d	E2h	â	242d	F2h	ò
131d	83h	ƒ		147d	93h	“		163d	A3h	£	179d	B3h	³	195d	C3h	Ã	211d	D3h	Ó	227d	E3h	ã	243d	F3h	ó
132d	84h	„		148d	94h	”		164d	A4h	¤	180d	B4h	´	196d	C4h	Ä	212d	D4h	Ô	228d	E4h	ä	244d	F4h	ô
133d	85h	…		149d	95h	•		165d	A5h	¥	181d	B5h	µ	197d	C5h	Å	213d	D5h	Õ	229d	E5h	å	245d	F5h	õ
134d	86h	†		150d	96h	–		166d	A6h	¦	182d	B6h	¶	198d	C6h	Æ	214d	D6h	Ö	230d	E6h	æ	246d	F6h	ö
135d	87h	‡		151d	97h	—		167d	A7h	§	183d	B7h	·	199d	C7h	Ç	215d	D7h	×	231d	E7h	ç	247d	F7h	÷
136d	88h	ˆ		152d	98h	˜		168d	A8h	¨	184d	B8h	¸	200d	C8h	È	216d	D8h	Ø	232d	E8h	è	248d	F8h	ø
137d	89h	‰		153d	99h	™		169d	A9h	©	185d	B9h	¹	201d	C9h	É	217d	D9h	Ù	233d	E9h	é	249d	F9h	ù
138d	8Ah	Š		154d	9Ah	š		170d	AAh	ª	186d	BAh	º	202d	CAh	Ê	218d	DAh	Ú	234d	EAh	ê	250d	FAh	ú
139d	8Bh	‹		155d	9Bh	›		171d	ABh	«	187d	BBh	»	203d	CBh	Ë	219d	DBh	Û	235d	EBh	ë	251d	FBh	û
140d	8Ch	Œ		156d	9Ch	œ		172d	ACh	¬	188d	BCh	¼	204d	CCh	Ì	220d	DCh	Ü	236d	ECh	ì	252d	FCh	ü
141d	8Dh			157d	9Dh			173d	ADh		189d	BDh	½	205d	CDh	Í	221d	DDh	Ý	237d	EDh	í	253d	FDh	ý
142d	8Eh	Ž		158d	9Eh	ž		174d	AEh	®	190d	BEh	¾	206d	CEh	Î	222d	DEh	Þ	238d	EEh	î	254d	FEh	þ
143d	8Fh			159d	9Fh	Ÿ		175d	AFh	¯	191d	BFh	¿	207d	CFh	Ï	223d	DFh	ß	239d	EFh	ï	255d	FFh	ÿ

Hexadecimal to Binary

0	0000	4	0100	8	1000	C	1100
1	0001	5	0101	9	1001	D	1101
2	0010	6	0110	A	1010	E	1110
3	0011	7	0111	B	1011	F	1111

Groups of ASCII-Code in Binary

Bit 6	Bit 5	Group
0	0	Control Characters
0	1	Digits and Punctuation
1	0	Upper Case and Special
1	1	Lower Case and Special