

# FUNDAMENTOS DE COMPUTADORES

## Práctica 8

### Ensamblador: Análisis de programas. Codificación de instrucciones.

Durante la realización de la práctica **en el laboratorio** deberás responder las preguntas de un **examen** de Poliformat. Es conveniente que visualices y resuelvas este examen al mismo tiempo que avanzas el trabajo de la práctica. Si estáis realizando la práctica en pareja tenéis dos opciones:

Opción a: primero realizáis el examen con uno de los usuarios, y una vez finalizado el trabajo y el examen, cambiáis de usuario y volvéis a realizar el examen.

Opción b: utilizáis dos navegadores diferentes al mismo tiempo, por ejemplo IE y Firefox. Esto os permite abrir dos sesiones diferentes en poliformat y responder los dos exámenes al mismo tiempo.

Es **importante que resuelvas antes de acudir** al laboratorio los ejercicios que se plantean en este boletín, ya que las preguntas del examen están muy relacionadas con ellos, y no tendrás tiempo para hacerlo todo en el laboratorio.

### INTRODUCCIÓN Y OBJETIVOS

En esta última práctica se vuelve a hacer uso del simulador PCSpim para la ejecución de programas en ensamblador. Los objetivos que se persiguen son los siguientes:

- Analizar programas con estructura iterativa, que incluyen instrucciones de salto condicional e incondicional.
- Estudiar programas que procesan cadenas de caracteres.
- Estudiar programas que realizan cálculos matemáticos iterativos sencillos, manejando datos numéricos enteros en memoria.
- Analizar la codificación de instrucciones de diferente tipo.

### TRUNCAMIENTO DE CADENA DE CARACTERES

El código que se va a emplear en las siguientes preguntas es el siguiente. Escríbalo en un archivo y llámelo “practica\_mips\_1.s”, cárguelo en el PCSPIM y compruebe que se carga correctamente.

```
.data 0x10000000
ini:  .asciiz "Hola"
sel:  .word 3
res:  .space 4
tot:  .word -1

.text 0x00400000
.globl __start
__start:
    la $8, ini
    la $9, sel
    lw $9, 0($9)
    la $10, res
    add $11, $0, $0
bucle:
    beq $9, $0, fin
    lb $12, 0($8)
    sb $12, 0($10)
    addi $8, $8, 1
    addi $9, $9, -1
    addi $10, $10, 1
    addi $11, $11, 1
    j bucle
fin:
    la $12, tot
    sw $11, 0($12)
.end
```

Código 1. Truncamiento de una cadena de caracteres ASCII (archivo “practica\_mips\_1.s”)

**Pregunta 1.** Indique el contenido del segmento de datos antes de iniciarse la ejecución, teniendo en cuenta que los datos se almacenan en formato “little endian”. El contenido debe ponerse en hexadecimal por cada byte de memoria.

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

**Pregunta 2.** Indique el contenido del segmento de datos una vez finalizada la ejecución, teniendo en cuenta que los datos se almacenan en formato “little endian”. El contenido debe ponerse en hexadecimal por cada byte de memoria.

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

**Pregunta 4.** Determinar el contenido de los siguientes registros cuando se haya ejecutado por primera vez la instrucción j bucle.

Registro	Contenido
\$8	
\$9	
\$10	
\$11	
\$12	

**Pregunta 3.** Determinar el contenido de los siguientes registros cuando se haya finalizado la ejecución del programa.

Registro	Contenido
\$8	
\$9	
\$10	
\$11	
\$12	

Escriba el código que aparece en el PCSPIM de las instrucciones que se detallan, tanto en hexadecimal como en binario, separando (en el caso del binario) cada campo.

**Pregunta 5.** Instrucción tipo R.

Instrucción:	add \$11, \$0, \$0	Código hexadecimal:			
Código binario:					
CO	rs	rt	rd	Numdesp	Función

**Pregunta 6.** Instrucción tipo I.

Instrucción:	lb \$12, 0(\$8)	Código hexadecimal:	
Código binario:			
CO	rs	rt	Desp/Inm

**Pregunta 7.** Localice en el PCSPIM los siguientes códigos de instrucciones y determine con qué instrucciones se corresponden:

Código	Instrucción
0x3c011000	
0x11200008	
0x08100007	

**Examen (si estas en el laboratorio):**

**Ahora es el momento de acudir a poliformat y responder las preguntas relacionadas con el programa “practica\_mips\_1.s”.**

## FUNCIONALIDAD DE PROGRAMAS

Escriba el siguiente código en un archivo “practica\_mips\_2.s”, cárguelo en el PCSPIM y compruebe que se carga correctamente. A partir de la ejecución del mismo, responda a las siguientes preguntas.

```
.globl __start
.data 0x10000000
vector:    .word 3,4,5
ncomp:     .word 3
result:    .space 4

.text 0x00400000
__start:
    la $8, vector
    la $9, ncomp
    lw $9, 0($9)
    addi $10, $0, 0
bucle:
    beq $9, $0, fin
    lw $11, 0($8)
    add $10, $10, $11
    addi $8, $8, 4
    addi $9, $9, -1
    j bucle
fin:
    la $12, result
    sw $10, 0($12)
.end
```

**Código 2. practica\_mips\_2.s**

**Pregunta 8.** De las siguientes funciones, determine cuál de ellas implementa el código anterior y describa cómo se ha llegado a esa respuesta, teniendo en cuenta que la posición del primer elemento de un vector comienza por 0.

$f = \sum_{i=0}^2 \text{vector}[i]$	$f = \sum_{i=0}^2 i + \text{vector}[i]$	$f = \sum_{i=0}^2 i \cdot \text{vector}[i]$
(a)	(b)	(c)

Código	Función
practica_mips_2.s	

Justificación de la respuesta



**Pregunta 10.** Indique la codificación en binario y hexadecimal de todas las instrucciones de salto que aparecen en el código “practica\_mips\_3.s”. Justifique su respuesta indicando a qué direcciones se encuentran asociadas las etiquetas de dichos saltos.

**Examen (si estas en el laboratorio):**

**Ahora es el momento de acudir a poliformat y responder las últimas preguntas del examen”**

TABLA ASCII

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	♦	(et)	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)	(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	ƙ
012d	0Ch	♂	(np)	028d	1Ch	L	(fs)	044d	2Ch	,	060d	3Ch	<	076d	4Ch	L	092d	5Ch	\	108d	6Ch	l	124d	7Ch	l
013d	0Dh	♂	(cr)	029d	1Dh	↵	(gs)	045d	2Dh	-	061d	3Dh	=	077d	4Dh	M	093d	5Dh	]	109d	6Dh	m	125d	7Dh	ƥ
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	ˆ	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	§	154d	9Ah	§	170d	AAh	ª	186d	BAh	º	202d	CAh	Ê	218d	DAh	Ú	234d	EAh	ê	250d	FAh	û
139d	8Bh	<	155d	9Bh	>	171d	ABh	«	187d	BBh	»	203d	CBh	Ë	219d	DBh	Û	235d	EBh	ë	251d	FBh	ü
140d	8Ch	£	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	AEnh	®	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	ÿ	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

© 2009 Michael Goerz  
 This work is licensed under the Creative Commons  
 Attribution-Noncommercial-Share Alike 3.0 License.  
 To view a copy of this license, visit  
<http://creativecommons.org/licenses/by-nc-sa/>