Ejercicios Examen EP2 – AO 2020/21

BLOQUE 1:

1.- A partir de la posición de memoria etiquetada como cadena se encuentra una secuencia de caracteres ASCII terminados en '?'. Realice un programa que cuente el número de caracteres que se corresponden con letras (mayúsculas o minúsculas) y números y deje el resultado en r0.

```
            .data
cadena:     .ascii      "& ep2 : ! aprobado ?"
```

Con el ejemplo anterior en el registro r0 debería quedar un 11

2.- Realice un programa que procese un número en BCD de ocho dígitos que se encuentra almacenado en la posición de memoria etiquetada como 'numbcd' y cacule su 'espejo', es decir, la posición más significativa pase a ser la menos significativa, la de peso 6 pase a ser la de peso 1 y así sucesivamente. El resultado deberá quedar almacenado en r0

```
            .data
numbdc      .word       0x12345678
```

Con los datos del ejemplo en r0 debe quedar 0x87654321.

NOTA: En sistemas de computación, Binary-Coded Decimal (BCD) o Decimal Codificado en Binario es un estándar para representar números decimales en el sistema binario, en donde cada dígito decimal es codificado con una secuencia de 4 bits. Así el 0 se presenta como 0000, el 1 como 0001… el 9 como 1001.

3.- Realizar un programa que dado un número de 64 bits almacenado en la posición de memoria etiquetada como 'num' cuente la cantidad de unos que tiene la representación binaria de ese número y deje el resultado en la palabra etiquetada como 'nunos'

```
            .data
nunos:      .space      4
num:        .quad       0x02AF785E01B59B34
```

Con los datos del ejemplo anterior en la posición etiquetada como 'nunos' debe quedar un 30

4.- Realice un programa que cuente el número de palabras de una cadena de caracteres ASCII terminada en cero y situada en la posición etiquetada como 'cadena'. Se considerará como palabra una sucesión de letras (mayúsculas o minúsculas) separadas por cualquier otro carácter ASCII. El resultado deberá almacenarse en la posición etiquetada como 'npal'

```
            .data
npal:       .space      4
cadena:     .asciz      "Una?dos_= tres   ."
```

Con los datos del ejemplo anterior, en la posición npal deberá quedar almacenado un 3

5.- Escribir un programa que dado un número natural representado por una cadena de caracteres ASCII que empieza en la posición etiquetada como ‘numero’ y terminada por un cero, indique si el número es capicúa. Si lo es dejará en la posición etiquetada como ‘capicua’ una ‘S’ y una ‘N’ en caso contrario. Si la cadena está vacía dejará una ‘V’.

```
            .data
capicua:    .ascii      "X"
numero:     .asciz      "12321"
```

Con los datos del ejemplo anterior, el programa deberá dejar una ‘S’ en la posición etiquetada como ‘capicua’

6.- Escribir un programa que dado un número natural representado por una cadena de caracteres ASCII que empieza en la posición etiquetada como ‘número’ y terminada por un cero, indique si el número es capicúa. Si lo es dejará en la posición etiquetada como capicua una ‘S’, en caso contrario dejará una ‘N’ en dicha posición y modificará la cadena ‘numero’ dejando una ‘X’ en todas aquellas posiciones que impiden que se cumpla la condición

```
            .data
capicua:    .ascii       "X"
numero:     .asciz       "19234281"
```

Con los datos del ejemplo anterior, el programa deberá dejar una ‘N’ en la posición etiquetada como capicua y modificará la cadena ‘numero’ dejando: “1X2XX2X1”.

BLOQUE 2:

1.- Realice un programa que identifique la presencia de una secuencia de caracteres en otra de mayor o igual tamaño. La cadena a buscar (puede considerar que nunca está vacía) está situada a partir de la posición de memoria etiquetada como ‘findme’ y su finalización la marca un ‘?‘. La cadena sobre la que se realizará la búsqueda se encuentra en la posición de memoria etiquetada como ‘searchin’ y su finalización está marcada por un cero. El programa devolverá en r0 un 1 (está) o un 0 (no está) dependiendo del resultado

```
            .data
findme:     .ascii      "estou?"
searchin:   .asciz      "   estou:pois claro que estou."
```

Con los datos del ejemplo anterior en r0 deberá quedar un 1

2.- En la posición de memoria etiquetada como ‘numbcd’ se encuentra almacenado un número expresado en BCD de ocho dígitos. Realice un programa que calcule su equivalente en decimal y lo deje en r0

```
            .data
numbdc      .word       0x12345678
```

Con los datos del ejemplo en r0 debe quedar un 12.345.678

3.- Realice un programa para calcular la representación en código ASCII del valor en hexadecimal almacenado en los 16 bits de la posición etiquetada como ‘num’ y deje el resultado en la posición etiquetada como ‘ascii’

```
            .data
ascii:      .space      4
num:        .hword      0xA30F
```

Con los datos del ejemplo en la posición etiquetada como ‘ascii’ debe quedar una “F”, en la siguiente un “0”, en la siguiente un “3” y en la siguiente una “A”

4.- Considere una zona de memoria, etiquetada como ‘zona’, que contiene una serie de números de 16 bits. Los números pueden ser tanto positivos como negativos o cero. La cantidad de números almacenados se guarda en la posición etiquetada como ‘nums’. Como mínimo habrá un número.

Realice un programa que deje en r0 el menor y en r1 el mayor de los números almacenados en ‘zona’ utilizando su valor absoluto como elemento de comparación. En caso de que aparezca tanto el número ‘x’ como el número ‘-x’ debe utilizarse -x para el menor y x para el mayor

```
      .data
zona: .hword          80, 76, -76, -128, 2, 128, -2
      .balign         4
nums  .word           7
```

Con los datos del ejemplo anterior, en r0 debe quedar ‘-2’ y en r1 ‘128’

5.- Realice un programa que cuente el número de apariciones de una secuencia de caracteres en otra de mayor o igual tamaño. La cadena a buscar (que nunca estará vacía) está situada a partir de la posición de memoria etiquetada como ‘findme’ y su finalización la marca un ‘!‘. La cadena sobre la que se realizará la búsqueda se encuentra en la posición de memoria etiquetada como ‘searchin’ y su finalización está marcada por un cero. El programa devolverá en r0 el número de apariciones.

```
            .data
findme:     ascii       "ama!"
searchin:   asciz       " amama  ama"
```

Con los datos del ejemplo anterior en r0 deberá quedar un 3

Tabla ASCII

Binario	*Dec*	*Hex*	*Representación*
0010 0000	32	20	espacio ( )
0010 0001	33	21	!
0010 0010	34	22	"
0010 0011	35	23	#
0010 0100	36	24	$
0010 0101	37	25	%
0010 0110	38	26	&
0010 0111	39	27	'
0010 1000	40	28	(
0010 1001	41	29	)
0010 1010	42	2A	*
0010 1011	43	2B	+
0010 1100	44	2C	,
0010 1101	45	2D	-
0010 1110	46	2E	.
0010 1111	47	2F	/
0011 0000	48	30	0
0011 0001	49	31	1
0011 0010	50	32	2
0011 0011	51	33	3
0011 0100	52	34	4
0011 0101	53	35	5
0011 0110	54	36	6
0011 0111	55	37	7
0011 1000	56	38	8
0011 1001	57	39	9
0011 1010	58	3A	:
0011 1011	59	3B	;
0011 1100	60	3C	<
0011 1101	61	3D	=
0011 1110	62	3E	>
0011 1111	63	3F	?

Binario	*Dec*	*Hex*	*Representación*
0100 0000	64	40	@
0100 0001	65	41	A
0100 0010	66	42	B
0100 0011	67	43	C
0100 0100	68	44	D
0100 0101	69	45	E
0100 0110	70	46	F
0100 0111	71	47	G
0100 1000	72	48	H
0100 1001	73	49	I
0100 1010	74	4A	J
0100 1011	75	4B	K
0100 1100	76	4C	L
0100 1101	77	4D	M
0100 1110	78	4E	N
0100 1111	79	4F	O
0101 0000	80	50	P
0101 0001	81	51	Q
0101 0010	82	52	R
0101 0011	83	53	S
0101 0100	84	54	T
0101 0101	85	55	U
0101 0110	86	56	V
0101 0111	87	57	W
0101 1000	88	58	X
0101 1001	89	59	Y
0101 1010	90	5A	Z
0101 1011	91	5B	[
0101 1100	92	5C	\
0101 1101	93	5D	]
0101 1110	94	5E	^
0101 1111	95	5F	_

Binario	*Dec*	*Hex*	*Representación*
0110 0000	96	60	`
0110 0001	97	61	a
0110 0010	98	62	b
0110 0011	99	63	c
0110 0100	100	64	d
0110 0101	101	65	e
0110 0110	102	66	f
0110 0111	103	67	g
0110 1000	104	68	h
0110 1001	105	69	i
0110 1010	106	6A	j
0110 1011	107	6B	k
0110 1100	108	6C	l
0110 1101	109	6D	m
0110 1110	110	6E	n
0110 1111	111	6F	o
0111 0000	112	70	p
0111 0001	113	71	q
0111 0010	114	72	r
0111 0011	115	73	s
0111 0100	116	74	t
0111 0101	117	75	u
0111 0110	118	76	v
0111 0111	119	77	w
0111 1000	120	78	x
0111 1001	121	79	y
0111 1010	122	7A	z
0111 1011	123	7B	{
0111 1100	124	7C	|
0111 1101	125	7D	}
0111 1110	126	7E	~