Grado en Ingeniería Informática - Ingeniería del Software
Estructura de Computadores - Prueba 2 - Junio 2018

*Apellidos, Nombre:*_____

Problema 1. Dado el programa escrito en lenguaje ensamblador del procesador CS2010, realice lo siguiente:

```
.EQU N=7

  LDI  R1,0
  LDI  R2,1
BUCLE:
  ST   (R1),R2
  ADDI R2,2
  ADDI R1,1
  CPI  R1,N
  BRLO BUCLE
  STOP
```

Dir	Contenido Memoria de Programa
00	11111 001 00000000
01	11111 010 00000001
02	00000 001 -----010
03	11000 010 00000010
04	11000 001 00000001
05	11011 001 00000111
06	00110 001 00000010
07	10111 --- --------
08	

Dir	Contenido Memoria de Datos
00	1
01	3
02	5
03	7
04	9
05	11
06	13
07	
08	

(a) Rellene el código máquina correspondiente al programa en la tabla *Contenido Memoria de Programa*. Rellene la tabla en **binario**. *(1 punto)*

(b) Rellene la tabla *Contenido Memoria de Datos* con los datos correspondientes tras la ejecución del programa. Rellene la tabla en **decimal**. *(1 punto)*

(c) Realice un programa equivalente escrito en lenguaje ensamblador de microcontroladores AVR, pero comenzando en la dirección de memoria 0x0100 en vez de la 0x00. *(1 punto)*

Los cambios a realizar son los siguientes: (1) LDI solo se puede realizar a partir del registro 16 (2) la escritura en memoria indirecta se realiza con X (2) ADDI no existe

```
  LDI  R16,0
  LDI  R17,1
  LDI  R18,2
  LDI  XH,0X01
  LDI  XL,0X00
BUCLE:
  ST   X+,R17
  ADD  R17,R18
  CPI  R17,N
  BRLO BUCLE
  BREAK
```

Problema 2. Indique el resultado final de los registros tras la ejecución de cada uno de los siguientes programas, escritos en lenguaje ensamblador para microcontroladores AVR.

(0.25 puntos)

```
CLR  R16
LDI  R17,0x44
ORI  R16,0x71
AND  R17,R16
BREAK
```

Valores finales en hexadecimal	
R16	0x71
R17	0x40

(0.25 puntos)

```
LDI  R16,0x01
LDI  R17,0x80
ADD  R17,R17
ADC  R16,R16
BREAK
```

Valores finales en hexadecimal	
R16	0x03
R17	0x00

(0.25 puntos)

```
LDI   R16,0x08
SBRS  R16,3
SUBI  R16,2
SUBI  R16,4
BREAK
```

Valor final en hexadecimal	
R16	0x04

(0.25 puntos)

```
EOR R16,R16
COM R16
    BREAK
```

Valor final en hexadecimal	
R16	0xFF

Problema 3. El programa mostrado pretende realizar el siguiente cálculo: Escribir en la dirección 0x0150 de la memoria RAM el resultado de sumar todos los bytes de un vector de tamaño N.

```
.include <m328def.inc>

.EQU N=8

  LDI  X,0x0150
  CLR  R1
  CLR  R16

bucle:
  LD   R0,X
  ADD  R1,R0
  ADDI X,1
  INC  R16
  BRNE bucle
  STS  0x100,R1

  break
```

(a) Indique las 2 instrucciones que no son correctas en lenguaje ensamblador de AVR. *(0.5 puntos)*

`LDI X,0x0150` y `ADDI X,1`

(b) Indique cual es la forma correcta de realizar estas instrucciones. *(0.5 puntos)*

```
LDI XH,0x01
LDI XL,0x50
```

`ADIW X,1` o `LD R0,X+`

(c) Además de las 2 instrucciones erróneas el bucle no está realizado correctamente. Proponga una solución correcta para el programa para que realice la operación solicitada. *(1 punto)*

```
.include <m328def.inc>

.EQU N=8

  LDI  XH,0x01
  LDI  XL,0x50
  CLR  R1
  CLR  R16

BUCLE:
  LD   R0,X+
  ADD  R1,R0
  INC  R16     ; También se puede cambiar INC R16 por DEC R16
  CPI  R16,N   ; si se pone DEC R16 no es necesario CPI R16,N
  BRNE BUCLE
  STS  0x100,R1

BREAK
```

(d) Otro posible problema es el desbordamiento en el resultado. Proponga una nueva solución donde el resultado sea de 16bits y no se produzca el desbordamiento. *(1 punto)*

```
Hay que utilizar 2 bytes para el resultado final y 2 registros para ir haciendo la
suma

.include <m328def.inc>

.EQU N=8

  LDI  XH,0x01
  LDI  XL,0x50
  CLR  R3 ; Este registro se usará para mantener un cero
  CLR  R2 ; Este registro será la parte alta del resultado → R2:R1
  CLR  R16

BUCLE:
  LD   R0,X+
  ADD  R1,R0
  ADC  R2,R3 ; Suma solo el carry ya que R3 es siempre cero
  INC  R16
  CPI  R16,N
  BRNE BUCLE
  STS  0x100,R2 ; El nuevo resultado es de 2 bytes, escribo primero la parte alta
  STS  0x101,R1

BREAK
```

Problema 4. Usando el lenguaje ensamblador de los microcontroladores AVR realice lo siguiente:

(a) Una subrutina que calcule una suma de 16 bits cumpliendo las siguientes especificaciones: *(1 punto)*

1. El primer número está almacenado en la dirección de memoria 0x0102 y 0x0103, estando la parte más significativa del número en la dirección de memoria 0x0102
2. El segundo número está almacenado en la dirección de memoria 0x0104 y 0x0105, estando la parte más significativa del número en la dirección de memoria 0x0104
3. El resultado se devolverá en los registros R1:R0, estando la parte más significativa en el registro R1
4. La subrutina no debe tener efectos laterales sobre el programa principal, es decir, todos los registros que se utilicen debe salvarse y recuperarse usando la pila del sistema.

```
SUBRUTINA:
  PUSH R2
  PUSH R3
  LDS  R1,0X0102
  LDS  R0,0X0103
  LDS  R3,0X0104   ; El número de 16bits será de la forma  R3:R2
  LDS  R2,0X0105
  ADD  R0,R2
  ADC  R3,R1
  POP  R3
  POP  R2
  RET
```

(b) Usando la subrutina anterior realice un programa capaz de sumar los números: 0x1234 + 0x3210. Los números están escritos en hexadecimal y el resultado debe almacenarse a partir de la dirección 0x0100 de la memoria. *(1 punto)*

```
LDI R16,0X12
STS 0X0102
LDI R16,0X34
STS 0X0103
LDI R16,0X32
STS 0X0104
LDI R16,0X10
```

```
STS 0X0105
CALL SUBRUTINA
STS 0X0100,R1
STS 0X0101,R0
BREAK
```

Problema 5. Realice un programa que configure los dos primeros pines del puerto C como salidas. Tras ello, el programa debe entrar en un bucle infinito generando una señal cuadrada en PINC0 y la inversa en PINC1, siguiendo las siguientes especificaciones: *(0.75 puntos)*

1. En el primer paso establecerá PORTC1=0V y PORTC0=5V **simultáneamente**.
2. En el segundo paso establecerá PORTC1=5V y PORTC0=0V **simultáneamente**.
3. Tras el paso anterior saltará incondicionalmente al paso 1.

```
.include <m328def.inc>

  LDI  R16,0x03
  OUT  DDRC,R16
  LDI  R16,0x01
  LDI  R17,0x02
BUCLE:
  OUT PORTC,R16
  OUT PORTC,R17
  JMP BUCLE
```

¿Cree que la señal medida en el osciloscopio será exactamente cuadrada?. Justifique su respuesta. *(0.25 puntos)*

```
No porque no todas las instrucciones tardan los mismos ciclos de reloj en
ejecutarse. JMP por ejemplo tarda 3 ciclos frente a 1 ciclo de cada OUT
```

***Nota**: El registro DDRC configura como entrada/salida el puerto C y el registro PORTC se utiliza para escribir en el puerto.*