INT. a los MICROPROCESADORES **NOMBRE:** _____
Parcial 1, OCTUBRE 2007 **CI:**_____

Cantidad de ejercicios: 8 **Cantidad total de puntos: 100**

El ejercicio 5 es de varias aseveraciones Verdadero - Falso.

Al final del parcial se debe entregar la solución de los ejercicios 1 al 5 marcada sobre las hojas de letra y la solución de los ejercicios restantes en hojas separadas por ejercicio escritas de un solo lado. Incluya su nombre y CI en cada hoja. Numere las hojas adicionales e indique el total de las mismas en la primera. Sea prolijo.

Ejercicio 1 (5 puntos)

Cuál es el contenido de los registros A y B al ejecutar la instrucción NOP en la rutina del recuadro?

```
        AND 00h
        LD A, 0Fh
        JP NZ, no_cero
cero:   LD B, 05h
        JP fin
no_cero: LD B, 06h
fin:    XOR B
        NOP
```

Hubo 4 variantes de esta pregunta:

Variante 0

A = 0AH 0000 1010 B

B = 5 0000 0101 B

Variante 1

```
        AND 00h
        LD A, 05h
        JP Z, cero
no_cero: LD B, 05h
        JP fin
cero:   LD B, 06h
fin:    XOR B
        NOP
```

A = 3 0000 0011 B

B = 6 0000 0110 B

Variante 2

```
        OR 02h
        LD A, 00h
        JP NZ, no_cero
cero:   LD B, 05h
        JP fin
no_cero: LD B, 06h
fin:    XOR B
        NOP
```

A = 6 0000 0110 B

B = 6 0000 0110 B

Variante 3

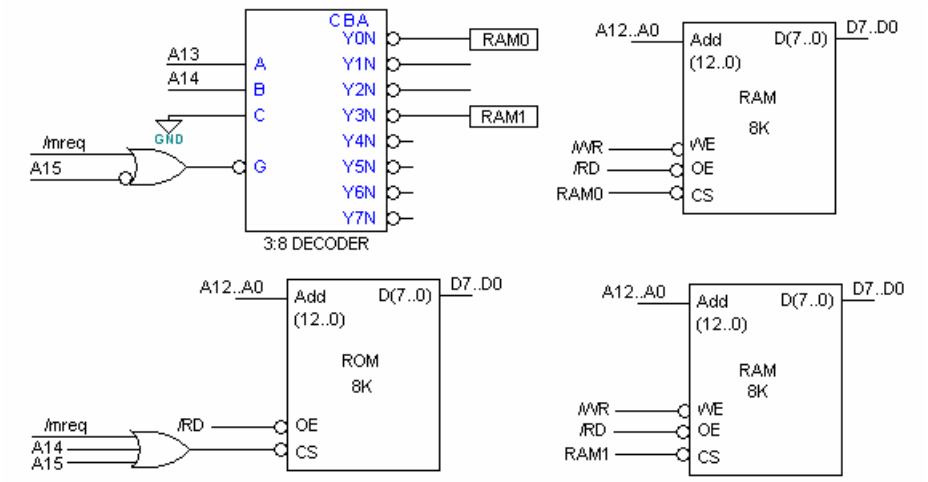
```
        OR 02h
        LD A, 0Fh
        JP NZ, no_cero
cero:   LD B, 05h
        JP fin
no_cero: LD B, 06h
fin:    XOR B
        NOP
```

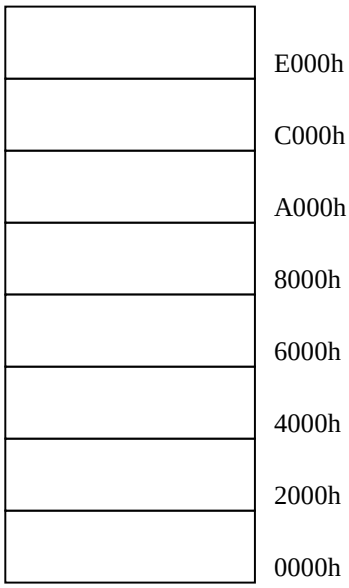
A = 9 0000 1001 B

B = 6 0000 0110 B

Ejercicio 2 (12 puntos)

Completar el siguiente mapa de memoria indicando qué chip se accede en cada dirección del espacio de memoria, dado el circuito de decodificación de la figura.
En las entradas del decodificador, C es la más significativa y A la menos significativa.
Los tres chips de memoria son de 8K x 8.





También hay cuatro variantes, la dibujada es la variante 0.

Variante 0:
* 0000h-1FFFh: ROM
* 2000h-3FFFh: ROM (FANTASMA)
* 4000h-5FFFh: VACIO
* 6000h-7FFFh: VACIO
* 8000h-9FFFh: RAM0
* A000h-BFFFh: VACIO
* C000h-DFFFh: VACIO
* E000h-FFFFh: RAM1

Variante 1: Variante 0 + AND a las salidas 2 y 3 de deco
* 0000h-1FFFh: ROM
* 2000h-3FFFh: ROM (FANTASMA)
* 4000h-5FFFh: VACIO
* 6000h-7FFFh: VACIO
* 8000h-9FFFh: RAM0
* A000h-BFFFh: VACIO
* C000h-DFFFh: RAM1 (FANTASMA)
* E000h-FFFFh: RAM1

Variante 2: Variante 1 + agrega A13 en Deco ROM
* 0000h-1FFFh: ROM
* 2000h-3FFFh: VACIO
* 4000h-5FFFh: VACIO
* 6000h-7FFFh: VACIO
* 8000h-9FFFh: RAM0
* A000h-BFFFh: VACIO
* C000h-DFFFh: RAM1 (FANTASMA)
* E000h-FFFFh: RAM1

Variante 3: Variante 0 + AND a las salidas 2 y 3 de deco y salidas 0 y 1
* 0000h-1FFFh: ROM
* 2000h-3FFFh: ROM (FANTASMA)
* 4000h-5FFFh: VACIO
* 6000h-7FFFh: VACIO
* 8000h-9FFFh: RAM0
* A000h-BFFFh: RAM0 (FANTASMA)
* C000h-DFFFh: RAM1 (FANTASMA)
* E000h-FFFFh: RAM1

Ejercicio 3 (13 puntos)

Dada la subrutina del recuadro ensamblarla completando la tabla de símbolos y las columnas “contador de posiciones”, “código de máquina” y “ciclos T”. No se insertan tiempos TW.

```
puerto1  EQU 01h
puerto2  EQU 02h

         ORG 8000h
DATO:    DB
TABLA:   DS 4

         ORG 2000h
         PUSH AF
         LD A, (TABLA)
         OUT (puerto1), A
         POP AF
         RET
```

Tabla de símbolos	
Símbolo	**Valor**
Puerto1	01
Puerto2	02
DATO	8000h
TABLA	8001h

Contador de posiciones	Código de máquina (hexadecimal)	Código de máquina (binario)	Instrucción	Ciclos T
2000	F5	11 110 101	PUSH AF	11
2001	3A	00 111 010	LD A,(TABLA)	13
	01	00000001		
	80	10000000		
2004	D3	11 010 011	OUT (puerto1), A	11
	01	00000001		
2006	F1	11 110 001	POP AF	10
2007	C9	11 001 001	RET	10

Ejercicio 4 (9 puntos)

Para las instrucciones LD y OUT de la subrutina del ejercicio anterior completar la información en la siguiente tabla indicando los ciclos de máquina que se ejecutan (M1, MEMRD, MEMWR, IORD, IOWR) y el valor que toman los buses de datos y de direcciones durante las transferencias de datos producidas en cada ciclo. Se supondrá que cuando se ejecuta la subrutina todos los lugares de memoria reservados para la variable tabla tienen almacenado el valor 27h.

Bus de direcciones	2001h	2002h	2003h	8001	2004	2005	xx01
Bus de datos	3A	01	80	27h	D3	01	27h
Tipo de ciclo	M1	MEMRD	MEMRD	MEMRD	M1	MEMRD	IOWR

Errores que aparecieron con frecuencia:

- No completar bus de direcciones
- No completar o completar parcialmente bus de datos
- Tabla de símbolos incompleta
- TABLA = 8000h en vez de 8001h
- Bytes ordenados al revés para 80 01 en el código

Ejercicio 5 (8 puntos)

Se desea realizar un programa que luego de un reset inicialice las variables VAR1 y VAR2 con 0 y 2 respectivamente. El sistema cuenta con 32K de ROM y 32K de RAM. Todos los programas están grabados en la ROM.

;Versión 1	***;Versión 2***	***;Versión 3***	***;Versión 4***
ORG 8000h	ORG 8000h	ORG 8000h	ORG 4000h
VAR1: DB	VAR1: DB	VAR1: DB	VAR1: DB
VAR2: DB	VAR2: DB	VAR2: DB	VAR2: DB
CERO: DB 0	CERO: EQU 0	CERO: EQU 0	CERO: EQU 0
DOS: DB 2	DOS: EQU 2	DOS: EQU 2	DOS: EQU 2
ORG 0000h	ORG 0000h	ORG 0000h	ORG 0000h
INICIA: LD A,(CERO)	INICIA: LD A,(CERO)	INICIA: LD A,CERO	INICIA: LD A,CERO
LD (VAR1), A	LD (VAR1), A	LD (VAR1), A	LD (VAR1), A
LD A,(DOS)	LD A,(DOS)	LD A,DOS	LD A,DOS
LD (VAR2), A	LD (VAR2), A	LD (VAR2), A	LD (VAR2), A
JP MAIN	JP MAIN	JP MAIN	JP MAIN

Para cada aseveración se debe responder si es verdadera (V) o falsa (F) encerrando con un círculo la letra que corresponda. Cada respuesta acertada vale los puntos indicados para la aseveración. Los puntos se restan si la respuesta es errada.

Variante 0

a) **(2 puntos)** La versión 1 es correcta V F ; *inicializa var. con DB*

b) **(2 puntos)** La versión 2 es correcta V F ; *usa ctes como direcc*

c) **(2 puntos)** La versión 3 es correcta V F

d) **(2 puntos)** La versión 4 es correcta V F ; *variables en ROM*

También había 4 variantes que eran combinaciones de:

- Las aseveraciones dicen “La versión xx está MAL” en lugar de “es correcta”. Las respuestas son complementarias de la anterior.
- Se intercambian las versiones 2 y 3 entre sí.

Ejercicio 6 (18 puntos)

El seudocódigo del recuadro recibe bytes por el puerto PUERTO_ENTRADA y los almacena en memoria RAM, hasta que el byte recibido sea FFh. En ese caso deberá retornar al programa principal devolviendo en el acumulador la cantidad de bytes recibidos.

La existencia de un nuevo byte para almacenar es notificada poniendo a 1 el bit 0 del puerto PUERTO_FLAG.

Escribir la subrutina RECEPCION que implemente el seudocódigo del recuadro. La dirección del PUERTO_FLAG es recibida en el registro H y la del PUERTO_ENTRADA en el registro L. El byte alto de la dirección de comienzo (BASE_HI) es recibido en D. No hace falta preservar registros. Debe devolverse en A la cantidad de bytes 00recibidos, se garantiza que nunca llegarán más de 250 bytes.

```
i=0
puntero = BASE_HI * 256

REPETIR
  MIENTRAS (bit 0 de PUERTO_FLAG = = 0)
    ESPERO
  FIN_MIENTRAS

  SI (byte en PUERTO_ENTRADA <> FFh)
    Memoria[puntero]=PUERTO_ENTRADA
    incremento i
    incremento puntero
  FIN_SI

HASTA PUERTO_ENTRADA = = FFh
A = i
retorno A
```

```
RECEPCION:
        LD B, 0         ; i= B =0
        LD E, 00h       ; puntero = DE = BASE_HI * 256
REPET:  LD C, H
FLAG:   IN A, (C)
        BIT 0,A
        JP Z, FLAG      ; MIENTRAS (bit 0 de PUERTO_FLAG = = 0), ESPERO

        LD C, L
        IN A, (C)
        CP A, 0FFh
        JP Z, RETORNO; SI (byte en PUERTO_ENTRADA <> FFh) entonces
        LD (DE), A              ; Memoria[puntero]= byte en PUERTO_ENTRADA
        INC B                   ; incremento i
        IND DE                  ; incremento puntero

        JP REPET        ; vuelvo a REPETIR
                        ; HASTA PUERTO_ENTRADA = = FFh
RETORNO:
        LD A, B         ; A = i
        RET
```

Errores que aparecieron con frecuencia:

- Suponer constantes a PUERTO_FLAG, PUERTO_ENTRADA o BASE_HI, se decía que se recibían en registros.
- Suponer que el contenido de los registros H y L se actualizan mágicamente con el valor del puerto. H y L contienen las direcciones PUERTO_FLAG y PUERTO_ENTRADA, los valores hay que leerlos con instrucciones IN
- Dificultades varias para armar el puntero BASE_HI * 256

Ejercicio 7 (15 puntos)

En diferentes partes de un programa se invoca a la subrutina **comparar** con secuencias de llamada como la del recuadro. La subrutina debe comparar dos trozos de memoria e indicar si contienen la misma información.
La rutina recibe a través del stack la dirección de comienzo del primer bloque, la dirección de comienzo del segundo bloque y el tamaño de los bloques a comparar (entre 1 y 255). Una vez comparados los bloques deberá devolver en el acumulador 00h si son iguales o FFh si hay alguna diferencia.
No se requiere que la subrutina preserve los registros.
Escribir la subrutina **comparar**.

```
pre:    ...
        LD BC, dir1
        PUSH BC
        LD BC, dir2
        PUSH BC
        LD A, tamaño
        PUSH AF
        CALL comparar
pos:    NOP
        ...
```

Si llamamos nnnn a la dirección hasta la que estaba ocupado el stack antes de la secuencia de llamada, después de la misma queda:

nnnn -->	///////////
nnnn - 1	dir1_H
nnnn - 2	dir1_L
	dir2_H
	dir2_L
	tamaño_H
	tamaño _L
	dir-ret_H
nNnnn - 8	**dir-ret_L**

```
comparar:
        pop ix   ; dirección de retorno
        pop bc   ; tamaño en B
        pop de   ; dir segundo bloque
        pop hl   ; dir primer bloque
        push ix  ; repongo dirección de retorno

for:
        ld a, (de)          ; leo un bloque
        cp (hl)             ; comparo con el otro
        jr nz, diferentes   ; si son diferentes salgo
        inc de              ; incremento punteros
        inc hl
        djnz for

iguales: ld a, 0            ; recorrí todo el bloque sin diferencias
        jr fin

diferentes:
        ld a, 0ffh

fin:    ret
```

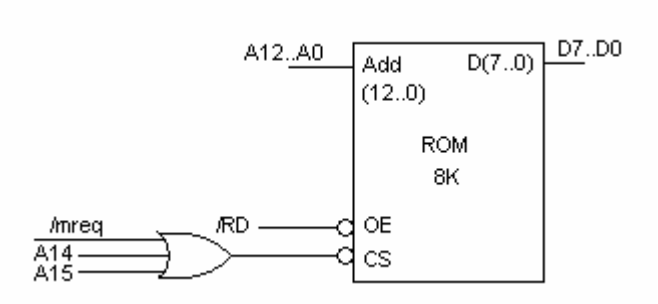
Errores que aparecieron con frecuencia:

- Ignorar la dirección de retorno en el stack
- Ignorar la dirección de retorno en el stack!
- Ignorar la dirección de retorno en el stack!!

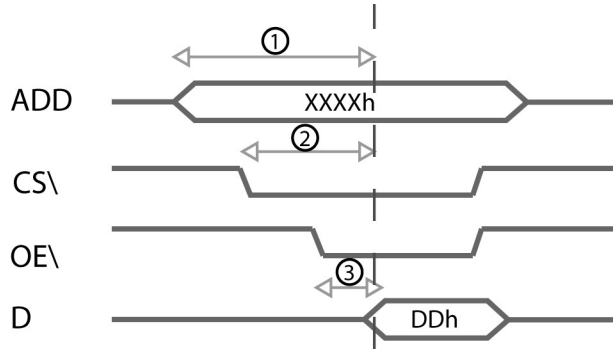
- No volver a alinear el stack. El programa principal no saca los parámetros del stack por lo que eso debe hacerlo la subrutina. De lo contrario en sucesivas invocaciones a la subrutina el stack iría creciendo hasta eventualmente salirse del área de RAM prevista para el mismo.

Ejercicio 8 (20 puntos)

Un sistema basado en un Z84C0010 (f_clk = 10MHz) utiliza una memoria ROM conectada como se indica en la figura. Indicar si es necesario insertar tiempos de espera TWAIT en el ciclo de lectura de memoria y en caso afirmativo indicar cuántos.
Plantear todas las ecuaciones necesarias indicando si los parámetros son máximos o mínimos antes de sustituir por valores numéricos.



El fabricante de la memoria indica que los tiempos característicos de la memoria son:



Tiempo	Min	Max
1- Tacc	--	285ns
2- Tcs	--	220ns
3- Toe	--	100ns

Los retardos de las compuertas (Tg) están entre 0 y 15ns
El reloj tiene ciclo de trabajo 50% (el tiempo en 1 es igual al tiempo en 0).

---

t25 = 25ns, t6_max = 65ns, t8_max = 55ns, t13_max = 65ns
Tacc
T1 + T2 + n Tw + T3H - t6_max - Tacc_max >= t25 req

n Tw >= t25 req(25) + t6_max(65) + Tacc_max(285) - T1(100) - T2(100) - T3H(50)
n Tw >= 375 - 250
n Tw >= 125 ==> n >= 2

Tcs
- por direcciones
T1 + T2 + n Tw + T3H - t6_max - tg_max(15) - Tcs_max(220) >= t25(25)
n Tw >= t25 req(25) + t6_max(65) + tg_max(15) + Tcs_max(220) - T1(100) - T2(100) - T3H(50)
n Tw >= 325 - 250
n Tw >= 75ns ==> n >= 1

- por MREQ
T1L + T2 + n Tw + T3H - t8_max(55) - tg_max(15) - Tcs_max(220) >= t25(25)
n Tw >= t25 req(25) + t8_max(55) + tg_max(15) + Tcs_max(220) - T1L(50) - T2(100) - T3H(50)
n Tw >= 315 - 200
n Tw >= 115ns ==> n >= 2

Toe
T1L + T2 + n Tw + T3H - t13_max(65) - Toe_max(100) >= t25(25)
n Tw >= t25(25) + t13_max(65) + Toe_max(100) - T1L - T2 - T3H
n Tw >= 190 – 200 ==> n >= 0

para cumplir todas las condiciones **hace falta n = 2 tiempos de espera**