Examen Final - Org. del Comp - 2008

2/3

1) Memory & I/O Mapped

IO/M	$A_{15}A_{14}A_{13}A_{12}$	$A_{11}A_{10}A_9A_8$	$A_7A_6A_5A_4$	$A_3A_2A_1A_0$	Memoria / Puerto
0	0 0 X X	. . . .	. . . .	. . . .	ROM (2732)
0	1 1 X .	. . . .	. . . .	. . . .	RAM (6264)
1	se repite la información de A_7-A_0 →		X X X X	X 0 . .	PPI (8255)
1			X X X X	X 1 X .	USART (8251)

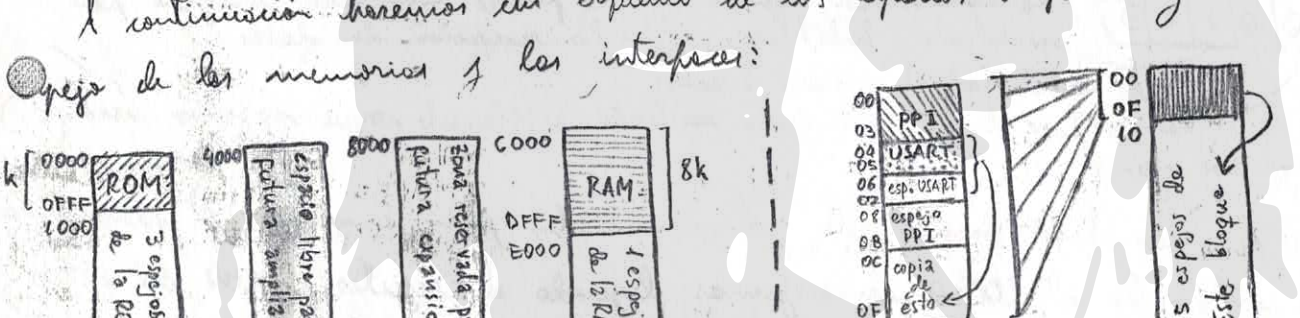
< habría que explicitar los espacios libres para ampliaciones,

Referencias: X condición sin cuidado (genera espejos)
. espacio reservado para la interfaz/memoria (o sea para acceder a sus datos internos)

Notemos que la ROM necesita 12 líneas para "manejo interno". Esto significa que el bloque de que disponemos (2732) tiene una capacidad de almacenamiento de 2^{12} = $= 4096 = 4 \times 1024 = 4k$, es decir que tenemos una EPROM $4k \times 8$

En cambio la RAM, que dicho sea de paso es estática, usa una línea más que la ROM. De esto se sigue que el bloque 6264 es una SRAM de $8k \times 8$

A continuación haremos un esquema de los espacios ocupados y las imágenes espejo de las memorias y las interfaces:



2) Lenguaje assembly (interpretación de las instrucciones)

Primero tenemos un salto incondicional hacia la 1ª línea de nuestro programa. Allí comenzamos inicializando el Stack Pointer donde corresponde (dirección de 16 bits "pointer") posiblemente en las últimas líneas de la RAM (DFFF)

Luego inicializamos el puerto paralelo cargando AO en el acumulador y enviando esto como "control word" a la dirección de su registro Control ("CTRLP"). Analicemos qué modalidad fue elegida: 'AO' = 1 0 1 0 0 0 0 0

- 1 → "elegimos el modo set"
- grupo A: modo 1 (1 0)
- 0 → registros A y C_H para salida
- grupo B: modo 0 (0)
- 0 → registros B y C_L para salida

A esto le sigue la preparación de los bits de PC (registro C del puerto paralelo) que será usado como control (para el handshake y las interrupciones) puesto que usamos el modo 1. La control

word seleccionado fue '0C' ≡ | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 0 |

"elegiremos el PC set/reset" ← sin cuidado; PC_6 → será reseteado

Como elegimos el modo 1 con grupo A → output ocurre que PC_6 hace las veces de $\overline{ACK}$, el cual habilita/deshabilita las interrupciones por parte de PA (el registro A del puerto paralelo, que es el encargado de recibir datos del micro y sacarlos hacia el puerto). Como $PC_6 = 0$ y $\overline{ACK}$ es activo por bajo ⇒ las interrupciones están habilitadas. Esto conforma el sist. de handshake.

A continuación iniciamos la interfaz 8251. Recordemos que luego de un reset podemos acceder por único reg al modo; luego sólo podremos modificar los comandos. Entonces el modo estará dado por la 1ª palabra que mandamos a su registro control: '5E' ≡ | 0 | 1 | 0 | 1 | 1 | 1 | 1 | 0 |

- 1 solo bit de parada
- paridad odd (impar)
- paridad habilit.
- datos de 8 bits de longitud
- baudrate : % 16 ⇒ #(baudios) = $\frac{R_xC}{16}$

Y le sigue a esto la elección del comando, pero el que elegimos la palabra '04' ≡ ...

... | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 |

- transmisión deshabilitada
- recepción habilitada

} el puerto serie será utilizado para la recepción de datos

- SBRK deshabilitado (no habrá breacks)
- los banderas de error no serán reseteadas, posiblemente porque no serán usadas

se deshabilitan el EH, RTS y IR (no habrá interrupciones internas)

Ahora sí, con todo inicializado estamos listos para empezar el programa propiamente dicho. En el LOOP0 comenzamos leyendo el registro STATUS del 8251 y lo comparamos con '02' → &

```
  - - - -   - - X -
  0 0 0 0   0 0 1 0
  ---------------
  0 0 0 0   0 0 X 0
```

} si X = 0 ⇒ $\overline{R_xRDY}$ ⇒ el receptor aún no está lleno, habrá que esperar. Pero esperar repetimos esta operación (la bandera Z estará reseteada pues el resultado fue 00h ⇒ el salto jz no retornará a LOOP0) si X = 1 ⇒ el receptor está libre y por ende listo (R_xRDY) ⇒ ya podemos continuar. Como Z = 0 ⇒ no volveremos a LOOP0. Entonces leemos el registro de recepción de la USART metiéndolo en el acumulador, y luego con DAA pasamos esa información a la codificación BCD natural. Terminamos guardando esto en la pila junto con las banderas (recuerda que sólo podemos spilear pares de registros en el SP del 8085, por ello para guardar el A hay que adosarle el PSW).

En el LOOP1 comenzamos leyendo el estado del puerto paralelo, el cual nos lo indica el registro C de la PPI (de ahora en más PC). Lo comparamos luego con 08:

```
  - - - -   X - - -
  0 0 0 0   1 0 0 0
  ---------------
  0 0 0 0   X 0 0 0
```

} si X = PC_3 = 0 ⇒ $\overline{INTR_A}$ (aún no se solicitó interrupción) ⇒ $\overline{OBF_A}$ (aún no se transmitió todo el output buffer). O sea que todavía no se descargó todo lo info del ciclo anterior desde el PPI al puerto ⇒ tenemos que esperar. Si

iguales a los del LOOP0. Si $PC_3 = X = 1$ ⇒ INTRA como consecuencia de OBF_A. En otras palabras, todo la información que estaba en PA (registro A del 8255) del ciclo anterior ya fue transmitida al puerto. Luego podemos cargar nueva información en PA para una nueva transmisión. Como jz no tendrá efecto esto nos recuperamos la información almacenada en la pila ("POP PSW") y la mandamos al PA para que sea enviada al puerto. Finalmente regresamos incondicionalmente al comienzo del 1° ciclo para esperar nuevos datos entrantes y repetir este proceso.

Así es que, en resumen, lo que hace este programa es recibir información por el puerto serie, transformarlo a codificación BCD natural y luego transmitirlo por el puerto paralelo. Notemos que sólo hay handshake: no hay interrupciones ni señal hlt ⇒ una vez que iniciamos esto la computadora continuará con el programa hasta que se le corte la alimentación o reciba algún tipo de reseteo.

3) Velocidad de recepción

Si queremos que los datos ingresen a la USART a 110 baudios, recordemos que al elegir el modo de esta interfaz elegimos 5E ⇒ baudaje [1|0] ó %16. Entonces como $\#(\text{baudios}) = \frac{R \times C}{[\text{modo baud}]} \Leftrightarrow 110^{\text{bauds}} = \frac{R \times C}{16} \Leftrightarrow 1760\ Hz = R \times C$. O sea que el reloj de recepción tendrá una velocidad de 1760 hercios.

4) Lenguaje de máquina

Como no hay nada especificado al respecto supondré que INICI está en los primeros lugares de la ROM, y que el salto se hacia 0040h habiendo dejado suficiente lugar para los RST por software y por hardware (o sea, libres 0000 - 003F)

Adress (h)	Nemónico		Lenguaje máquina (h)		
0000 - 0002	INICI:	JMP PROGR	C3	04	00
0040 - 0042	PROGR:	LXI SP, POINT	31	FF	DF
0043 - 0044	PPi:	MVI A, A0	3E	A0	
0045 - 0046		OUT CTRLP	D3	03	
0047 - 0048		MVI A, 0C	3E	0C	
0049 - 004A		OUT CTRLP	D3	03	
004B - 004C	USART:	MVI A, 5E	3E	5E	
004D - 004E		OUT CTRLS	D3	05	
004F - 0050		MVI A, 04	3E	04	
0051 - 0052		OUT CTRLS	D3	05	

(continúa al dorso)

Adress (h)	Nemónico		Lenguaje máquina (h)		
0053 - 0054	LOOP0 :	IN STATUS	DB	05	
0055 - 0056		ANI 02	E6	02	
0056 - 0058		JZ LOOP0	CA	53	00
0059 - 005A		IN RxD	DB	04	
005B		DAA	27		
005C		PUSH PSW	F5		
005D - 005E	LOOP1 :	IN PC	DB	02	
005F - 0060		ANI 08	E6	08	
0061 - 0063		JZ LOOP1	CA	5D	00
0064		POP PSW	F1		
0065 - 0066		OUT PA	D3	00	
0067 - 0069		JMP LOOP0	C3	53	00

Budde, Carlos E.