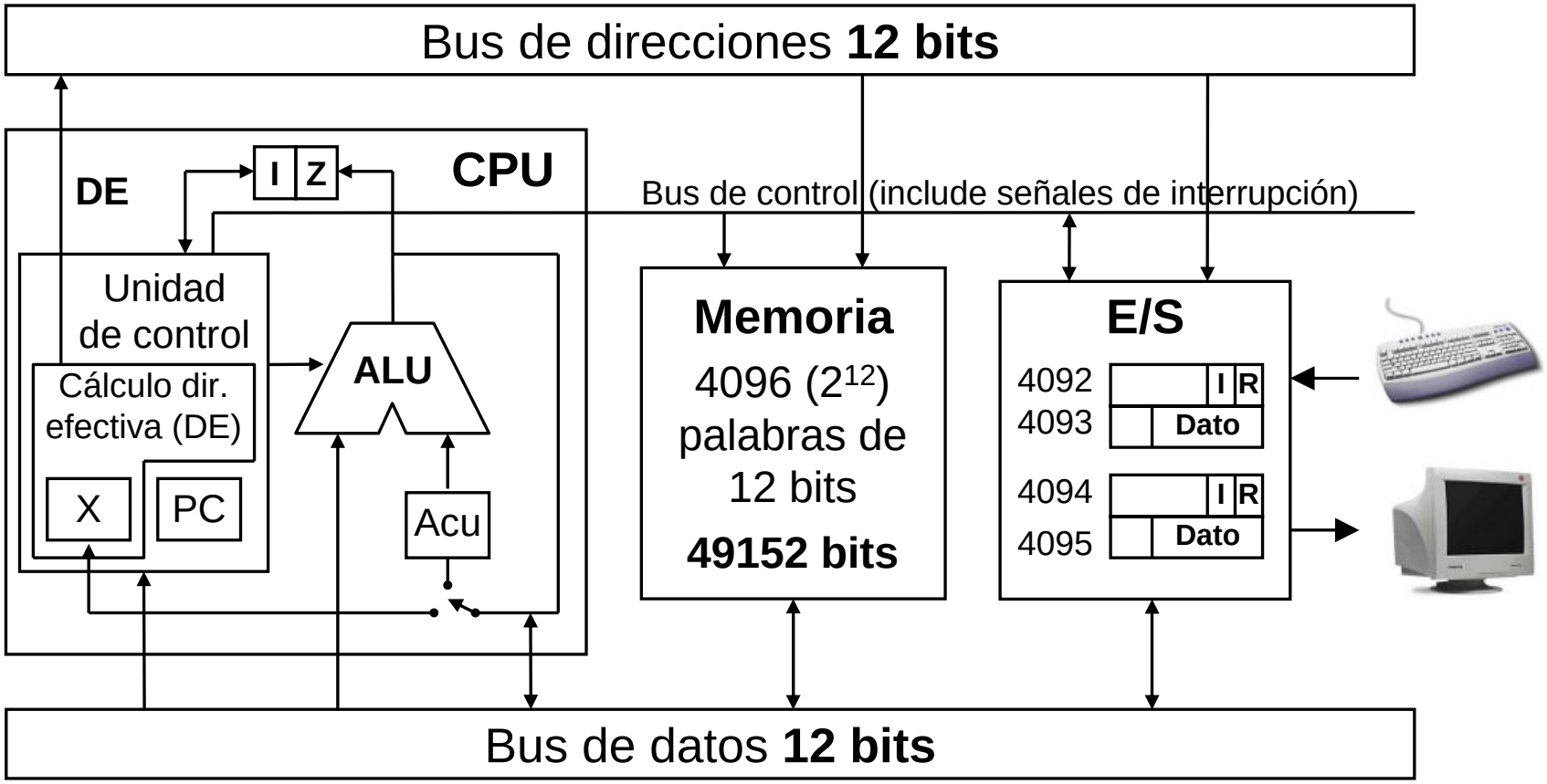
Simplez+i4

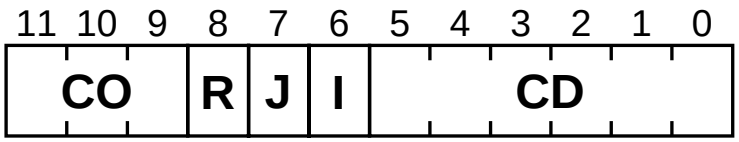
Una evolución del procesador Simplez

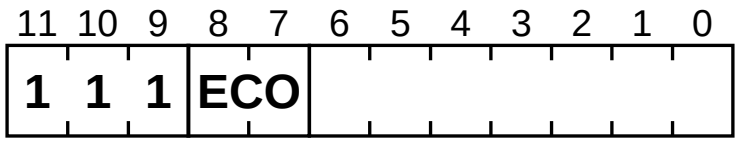
- Modos de direccionamiento indexado e indirecto.
- Mecanismo de interrupción para el manejo de periféricos.
- Rutinas de servicio de interrupciones.

Simplez+i4: modelo estructural



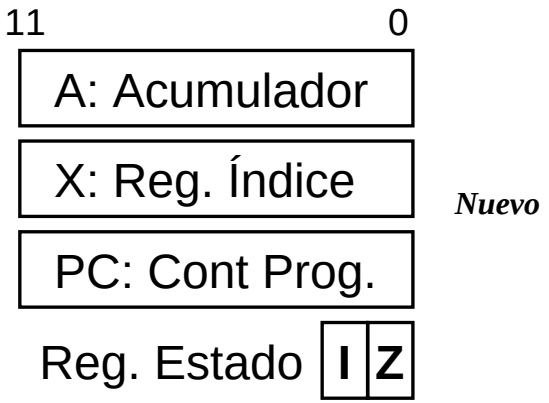
Formato y repertorio de instrucciones





CO ECO	Nemo.	Significado	
000	**ST**	(reg) → MP[DE]	
001	**LD**	(MP[DE]) → reg	
010	**ADD**	(reg) + (MP[DE]) → reg	
011	**BR**	DE → PC	
100	**BZ**	Si **Z**==1 DE → PC	
101	**LD #**	(CD) → reg	*Nuevo*
110	**SUB #**	(reg) – (CD) → reg	*Nuevo*
111 00	**HALT**	Detiene procesador	
111 01	**EI**	Permite interrupciones	*Nuevo*
111 10	**DI**	Inhibe interrupciones	*Nuevo*
111 11	**RTI**	Retorna de interrupción	*Nuevo*

Registros del microprocesador



(ver siguiente hoja)

Cálculo de la dirección efectiva (DE)

también llamada *efective address* (EA)

La dirección efectiva solo se usa en las instrucciones LD, ST, ADD, BR y BZ

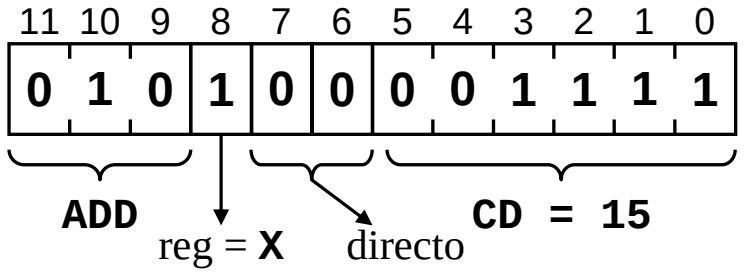
Modos de direccionamiento

J I	Modo	Cálculo de DE
00	Directo	DE = (CD)
01	Indirecto	DE = (MP[CD])
10	Indexado	DE = (CD)+(X)
11	Indirecto indexado	DE = (MP[CD])+(X)

Atención: A diferencia de Simplez, en Simplez+i4 las instrucciones BR y BZ calculan la dirección efectiva (DE) y cargan dicha dirección en el registro contador de programa **PC**.

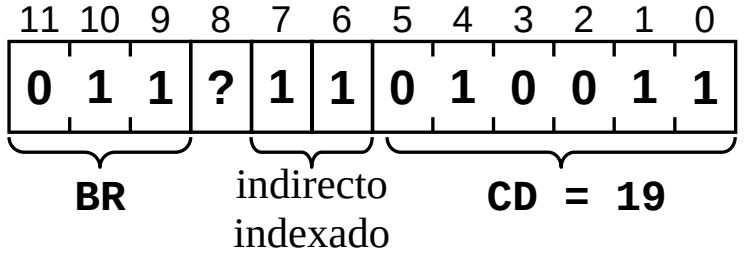
Atención: CD ≤ 63

Ejemplo de instrucción aritmética



(X) + (MP[15]) → X

Ejemplo de instrucción de salto



(MP[19]) + (X) → PC

Notación para los modos de direccionamiento

El direccionamiento inmediato solo se usa en las instrucciones `LD #` y `SUB #`

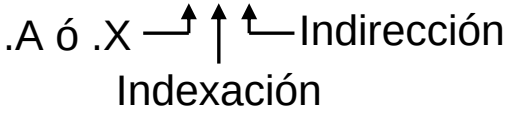
Ejemplos de notación:

Modo	Hex.	Lenguaje simbólico	Significado
Directo	`H'420`	`ADD .A,/32`	`A+ MP[32]⇨A`
Indirecto	`H'460`	`ADD .A,[/32]`	`A+ MP[MP[32]]⇨A`
Indexado	`H'4A0`	`ADD .A,/32[.X]`	`A+ MP[32+X]⇨A`
Indirecto indexado	`H'4E0`	`ADD .A,[/32][.X]`	`A+ MP[MP[32]+X]⇨A`
Inmediato	`H'C1B`	`SUB .A,#27`	`A – 27 ⇨A`

Atención: las instrucciones aritméticas modifican el estado del bit **Z** tanto para el registro accumulador (**A**) como para el registro índice (**X**)

Algunos ejemplos de modos de direccionamiento

Código en Binario	Hexad.	Código simbólico	Significado en notación RT
B'000 000 011011	H'01B	ST .A,/27	(A) → (MP[27])
B'001 100 111111	H'33F	LD .X,/63	(MP[63]) → X
B'010 001 100000	H'460	ADD .A,[/32]	(A)+(MP[(MP[32])]) → A
B'010 110 100000	H'5A0	ADD .X,/32[.X]	(X)+(MP[32+(X)]) → X
B'010 111 100000	H'5E0	ADD .X,[/32][.X]	(X)+(MP[(MP[32])+(X)])→X
B'011 000 010001	H'611	BR /17	17 → PC
B'011 001 010001	H'691	BR [/17]	(MP[17]) → PC
B'100 000 010101	H'815	BZ /21	Si Z==1: 21 → PC
B'100 011 001011	H'8CB	BZ [/11][.X]	Si Z==1: (MP[11])+(X) → PC
B'101 000 010101	H'A15	LD .A,#21	21 → A
B'110 100 000010	H'D02	SUB .X,#2	(X) - 2 → X
B'000 111 111111	H'1FF	ST .X,[/63][.X]	X → MP[(MP[63])+(X)]



Intercambiar zonas de memoria

(un programa de ejemplo para Simplez+i4)

Programa en lenguaje C

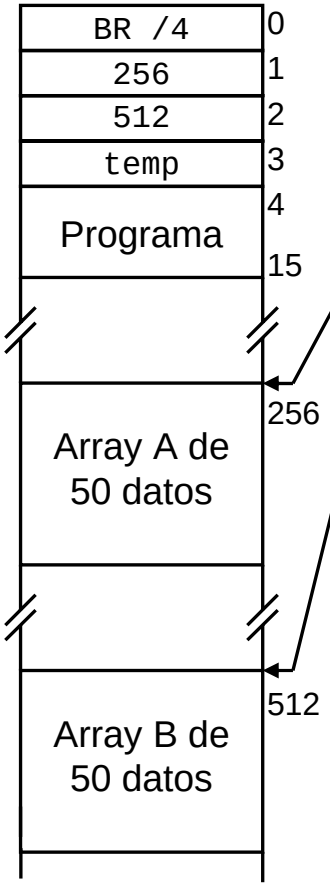
```
int temp;
int A[50];
int B[50];
for (c=0;c<50;c++)
{
  temp = A[c];
  A[c] = B[c];
  B[c] = temp;
}
```

Utilizaremos el registro **X** para llevar el contador `c`.

MP[1]: puntero al origen de `A`

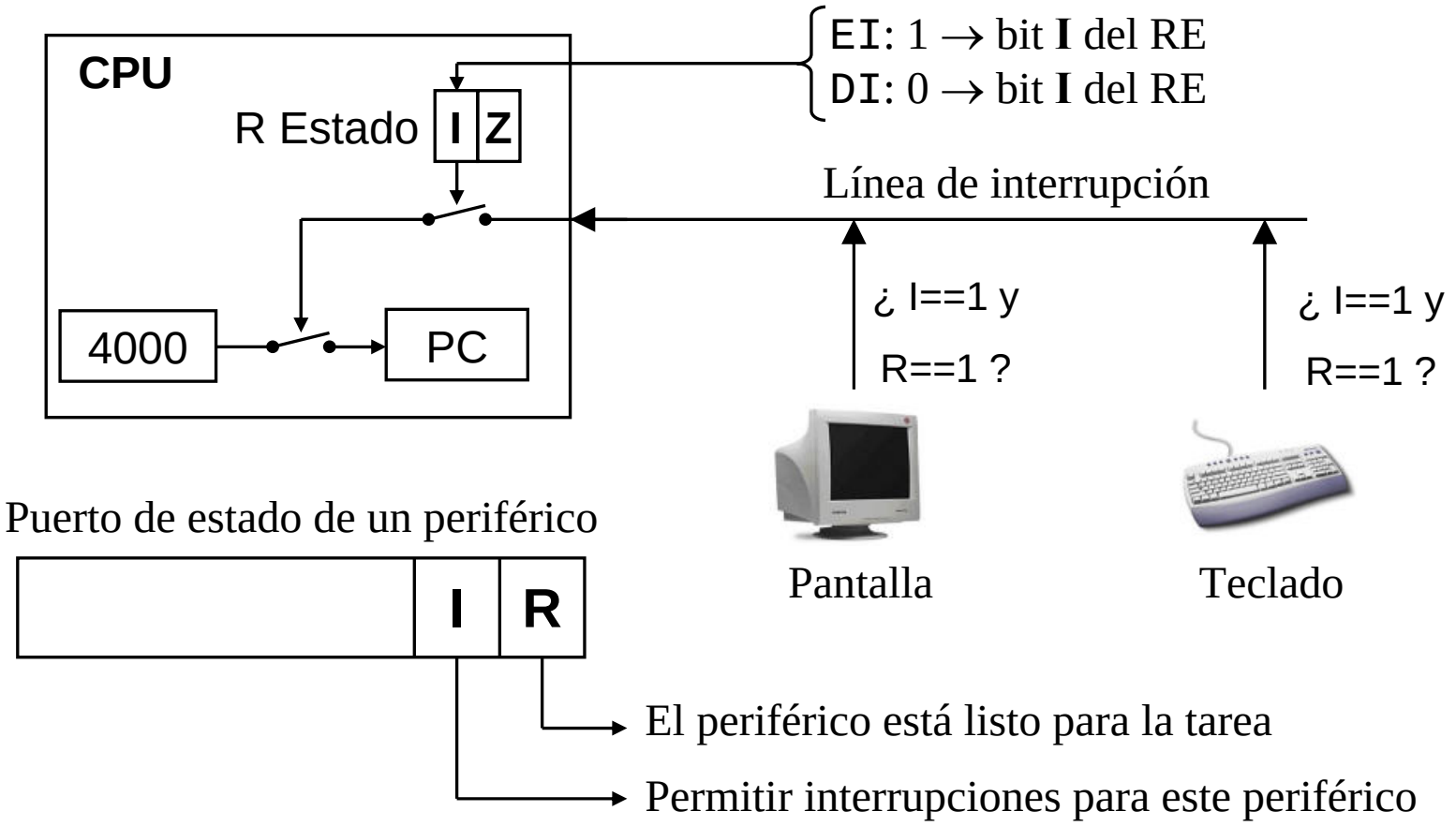
MP[2]: puntero al origen de `B`

MP[3]: variable `temp`



Dir	Cnt. Hex	C. Simbólico	
0	H'20A	BR	/4
1	H'100	256	
2	H'200	512	
3	H'000	0	
4	H'B31	LD	.X,#49
5	H'2C1	LD	.A,[/1][.X]
6	H'003	ST	.A,/3
7	H'2C2	LD	.A,[/2][.X]
8	H'0C1	ST	.A,[/1][.X]
9	H'203	LD	.A,/3
10	H'0C2	ST	.A,[/2][.X]
11	H'D00	SUB	.X,#0
12	H'80F	BZ	/15
13	H'D01	SUB	.X,#1
14	H'605	BR	/5
15	H'E00	HALT	

Interrupciones en Simplez+i4



La lectura del puerto de estado de un periférico termina una interrupción pendiente

Interrupciones: modelo procesal

Cuando aparece una interrupción y el bit **I** del registro de estado (RE) esta a 1:

La CPU, por hardware:

1.- Completa la instrucción en curso y calcula siguiente PC.
2.- Inhibe todas las interrupciones: 0 → bit **I** del RE
3.- Guarda el contenido de PC en MP[63]: PC → MP[63]
4.- Guarda el contenido del bit **Z** en MP[62]: bit **Z** → MP[62]
5.- Salta a 4000, donde se encuentra la rutina de servicio de la int. (ISR): 4000 → PC

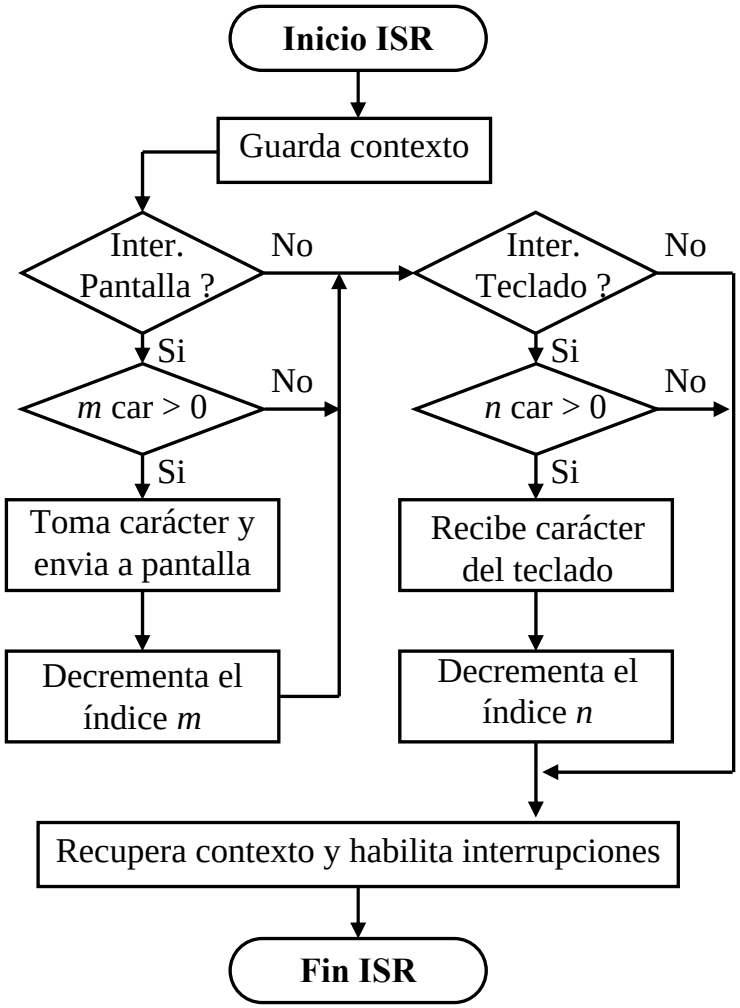
La rutina de servicio de la interrupción (ISR), por software:

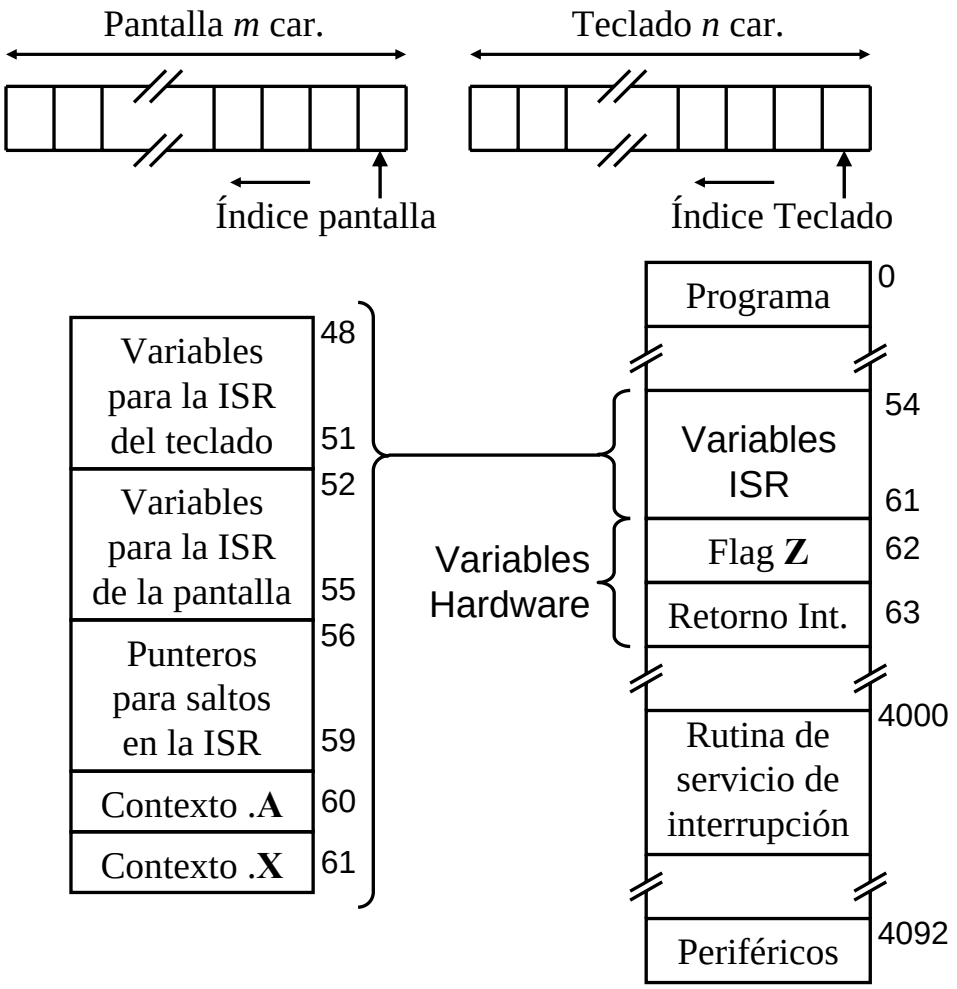
5.- Guarda el contexto de ejecución del programa (almacena valor de registros **X** y **A**).
6.- Atiende a los datos del periférico o los periféricos que han generado interrupción.
7.- Restaura el contexto de ejecución del programa (recupera valor de registros **X** y **A**).
8.- Permite interrupciones con `EI` (tiene efecto una instrucción más tarde. Sin reentrada)
9.- Retorna de la interrupción mediante `RTI`

Cuando se ejecuta `RTI`:

10.- Se recupera el estado bit **Z** almacenado previamente en MP[62]: (MP[62]) → bit **Z**
11.- Se retorna a la posición de ejecución almacenada en MP[63]: (MP[63]) → PC

Rutina de servicio de interrupción: estructura general





Rutina de servicio de interrupción (ISR): código

Rutina de servicio de interrupción (ISR)

```
4000 ST   .A,/60   ; Guarda
4001 ST   .X,/61   ; contexto
4002 LD   .A,[/52] ; Si int
4003 SUB  .A,#3    ; pantalla
4004 BZ   [/56]    ; haz ISR
4005 LD   .A,[/48] ; Si int
4006 SUB  .A,#3    ; teclado
4007 BZ   [/57]    ; haz ISR
4008 LD   .A,/60   ; Recupera
4009 LD   .X,/61   ; contexto
4010 EI   ; permite interrup.
4011 RTI  ; Ret prg principal
```

ISR de la pantalla

```
4012 LD   .X,/54    ; Carga contador pant.
4013 BZ   [/58]     ; Si contador==0 sal
4014 SUB  .X,#1     ; Decrementa contador
4015 LD   .A,[/55][.X]; Coje dato array y
4016 ST   .A,[/53]  ; mándalo a pantalla
4017 ST   .X,/54    ; Guarda contador pant.
4018 BR   [/58]     ; Retorna ISR principal
```

ISR del teclado

```
4019 LD   .X,/50    ; Carga contador tecl.
4020 BZ   [/59]     ; Si contador==0 salir
4021 SUB  .X,#1     ; Decrementa contador
4022 LD   .A,[/49]  ; Coge dato teclado y
4023 ST   .A,[/51][.X]; guárdalo en array
4024 ST   .X,/50    ; Guarda contador tecl.
4025 BR   [/59]     ; Retorna ISR principal
```

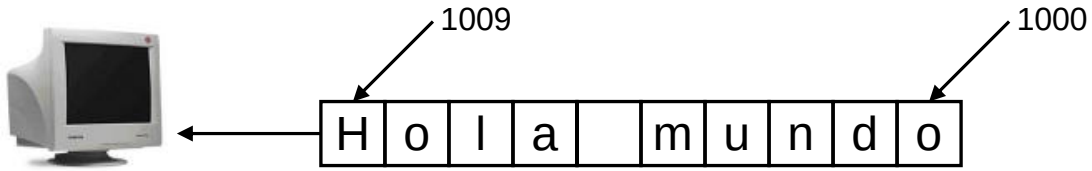
Ptr estado teclado	D’4092	48
Ptr datos teclado	D’4093	49
n caract. teclado	???	50
Ptr array teclado	???	51

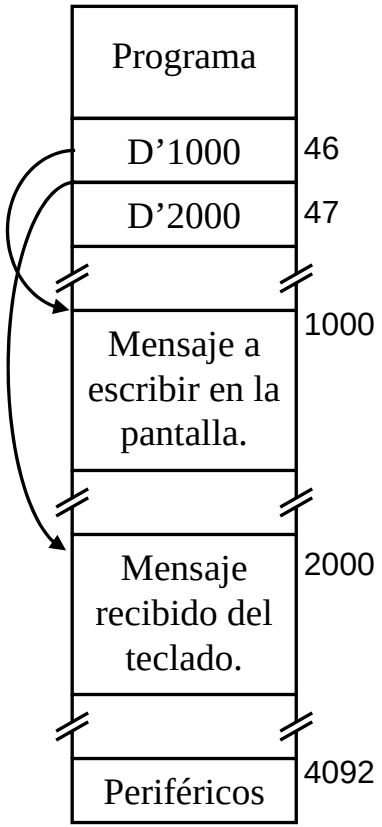
Ptr estado pantalla	D’4094	52
Ptr datos pantalla	D’4095	53
m caracteres pantalla	???	54
Ptr array pantalla	???	55

Ptr ISR pantalla	D’4012	56
Ptr ISR teclado	D'4019	57
Ptr continua ISR	D’4005	58
Ptr final ISR	D’4008	59

Interacción con la ISR de la pantalla

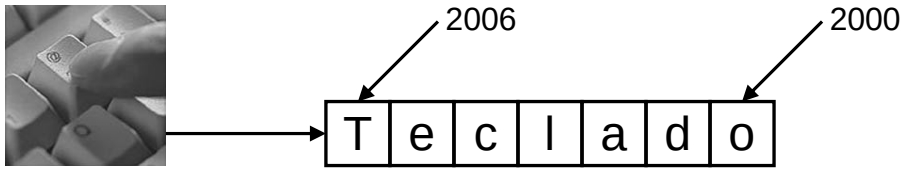
```
; Inicializamos los parámetros para imprimir una
; cadena de 10 caracteres almacenada en 1000
    ...
610  DI             ; Deshabilita interrupciones
611  LD   .A,/46    ; Coloca puntero al origen
612  ST   .A,/55    ; del array de pantalla
613  LD   .A,#10    ; Queremos poner 10 caracteres =
614  ST   .A,/54    ; valor inicial del contador.
615  LD   .A,#3     ; Inicia el puerto de estado de
616  ST   .A,[/52]  ; la pantalla permitiendo interrup.
617  EI             ; Habilita interrupciones
    ...
; Aquí podemos hacer cualquier tarea mientras los
; caracteres se van imprimiendo en la pantalla.
    ...
625  LD   .A,/54    ; Si queremos, podemos esperar a
626  BZ   /628      ; que se hayan impreso todos los
627  BR   /625      ; caracteres en la pantalla.
    ...
```

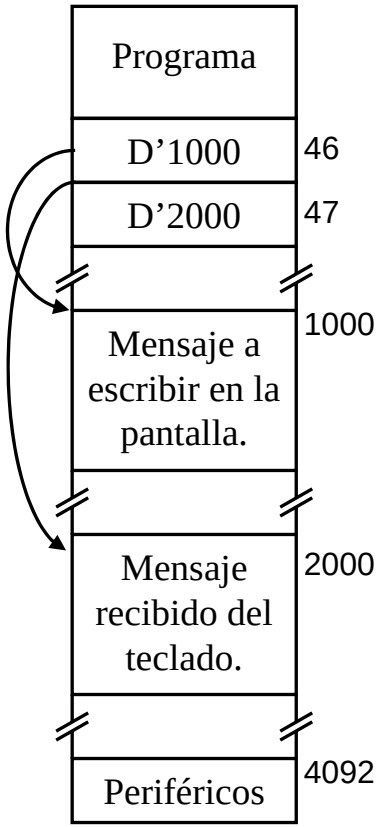




Interacción con la ISR del teclado

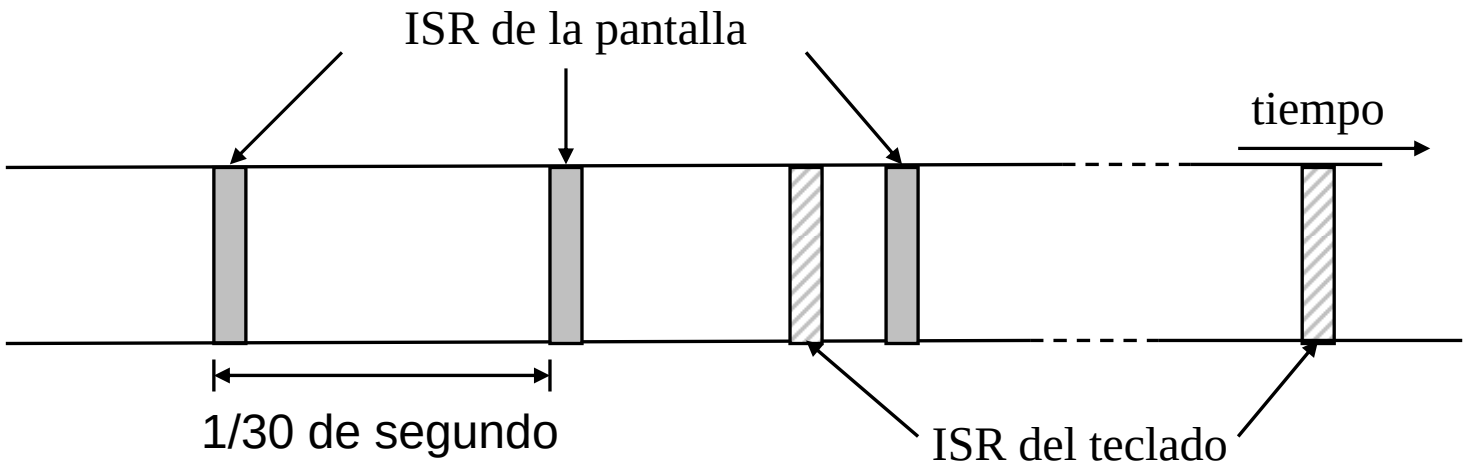
```
; Inicializamos los parámetros para recibir una
; cadena de 7 caracteres que se almacenará en 2000
    ...
630  DI             ; Deshabilita interrupciones
631  LD   .A,/47    ; Coloca puntero al origen del array
632  ST   .A,/51    ; de recepción del teclado.
633  LD   .A,#7     ; Esperamos recibir 7 carácteres =
634  ST   .A,/50    ; valor inicial del contador.
635  LD   .A,#2     ; Inicia el puerto de estado del
636  ST   .A,[/48]  ; teclado permitiendo ints
637  EI             ; Habilita interrupciones
    ...
; Aquí podemos hacer cualquier tarea mientras el
; usuario va escribiendo caracteres en el teclado.
    ...
641  LD   .A,/50    ; Si queremos, podemos esperar a
642  BZ   /644      ; que se hayan escrito todos los
643  BR   /641      ; caracteres esperados.
    ...
```





Concurrencia aparente

La ejecución del programa principal se ve interrumpida esporádicamente



La CPU “parece” que esta haciendo tres cosas al mismo tiempo:

- Ejecutar el programa principal.
- Enviar caracteres a la pantalla.
- Recibir caracteres del teclado.