Programación con Simplez

Conceptos y técnicas

- Programación de algoritmos de mayor complejidad.
- Modificación de instrucciones.
- Llamadas a subprogramas.
- Comunicaciones con periféricos. Método de espera activa.

Suma de *n* términos de una progresión aritmética (1/2)

$$res = \sum_{i=0}^{n} a_0 + i \cdot r$$

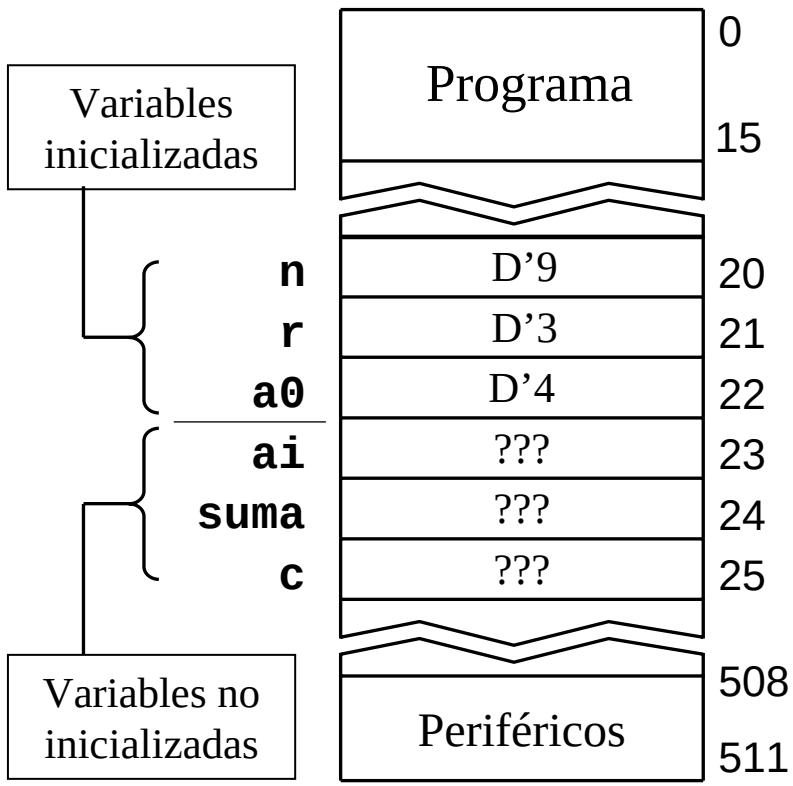
Programa en lenguaje de alto nivel (C)

```
ai = a0; suma = a0;
for (c = 1; c <= n; c++)
{
  ai = ai + r;
  suma = suma + ai;
}
```

Significado de las variables:

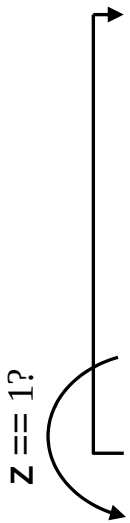
a0: primer término de la progresión ($i = 0$).
ai: término i-ésimo de la progresión.
suma: suma de los términos.
r: razón de la progresión aritmética.
c: variable que sirve de contador.

Asignación de direcciones
(y mapa de memoria antes de comenzar la ejecución)



Suma de *n* términos de una progresión aritmética (2/2)

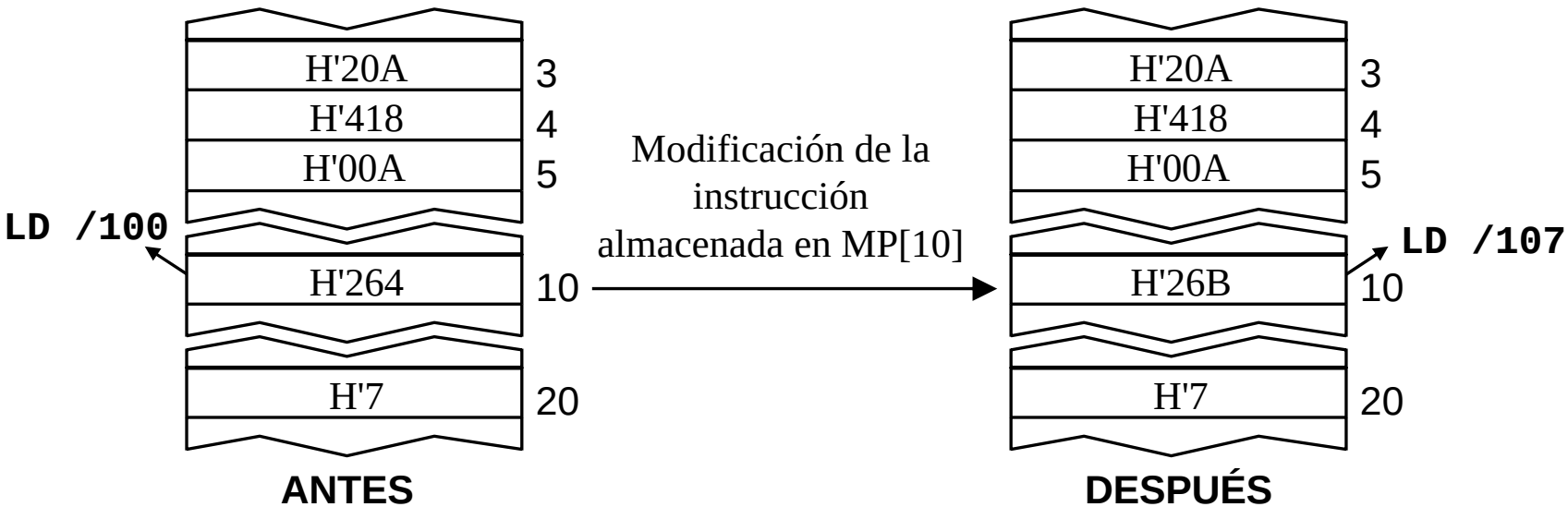
Dir MP	Cont. Binario	Cont. Hex	Cont. Oct.	Cont. Nem.		Comentarios
0	B'001000010110	H'216	O'1926	LD	/22	
1	B'000000010111	H'017	O'0027	ST	/23	; ai = a0
2	B'000000011000	H'018	O'0030	ST	/24	; suma = a0
3	B'001000010100	H'214	O'1024	LD	/20	
4	B'000000011001	H'019	O'0031	ST	/25	; c = n
5	B'001000010111	H'217	O'1027	LD	/23	
6	B'010000010101	H'415	O'2025	ADD	/21	
7	B'000000010111	H'017	O'0027	ST	/23	; ai = ai+r
8	B'010000011000	H'418	O'2030	ADD	/24	
9	B'000000011000	H'018	O'0030	ST	/24	; suma = suma+ai
10	B'001000011001	H'219	O'1031	LD	/25	
11	B'110000000000	H'C00	O'6000	DEC		; AC = contador-1
12	B'100000001111	H'80F	O'4017	BZ	#15	; Si 0, termina.
13	B'000000011001	H'019	O'0031	ST	/25	; contador = AC
14	B'011000000101	H'605	O'3005	BR	#5	
15	B'111000000000	H'E00	O'7000	HALT		



Modificación de instrucciones

Dir MP	Cont. Binario	Cont. Hex	Cont. Nem.	
…				
3	B'001000001010	H'20A	LD	/10
4	B'010000010100	H'418	ADD	/20
5	B'000000001010	H'00A	ST	/10
…				
10	B'001001100100	H'264	LD	/100
…				
20	B'000000000111	H'007		
…				

La "automodificación" de instrucciones es la única manera de recorrer zonas de memoria en Simplez.



Construcción de bucles para recorrer zonas de memoria

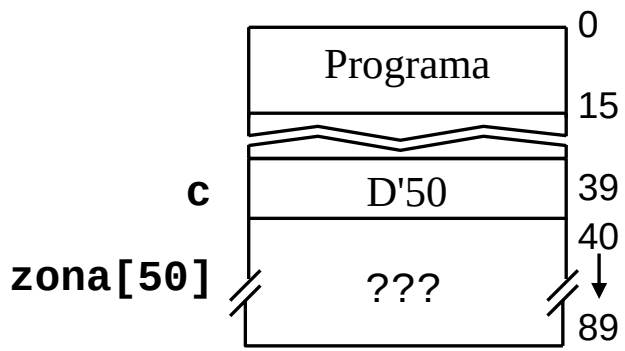
(ej. borrar un área de memoria)

Programa en lenguaje de alto nivel (C)

```
{ int c, zona[50];
  for (c=0; c<50; c++)
  { zona[c] = 0; }
}
```

Asignación de direcciones

(y mapa de memoria antes de comenzar la ejecución)




```
Dir   Cont. Hex  Cont. Nem.  Comentarios
0     H'A00      CLR        ; Borra celda
1     H'059      ST   /89    ; de memoria
2     H'201      LD   /1     ; Pasa a la
3     H'C00      DEC        ; siguiente
4     H'001      ST   /1     ; dirección
5     H'227      LD   /39    ; Decrementa
6     H'C00      DEC        ; el contador
7     H'80A      BZ   #10    ; del bucle
8     H'027      ST   /39    ; y termina si
9     H'600      BR   #0     ; ha llegado a
10    H'E00      HALT       ; cero.
```

MP[1] antes de ejecución	MP[1] después de ejecución
`ST /89`	`ST /39`

Subprogramas en Simplez

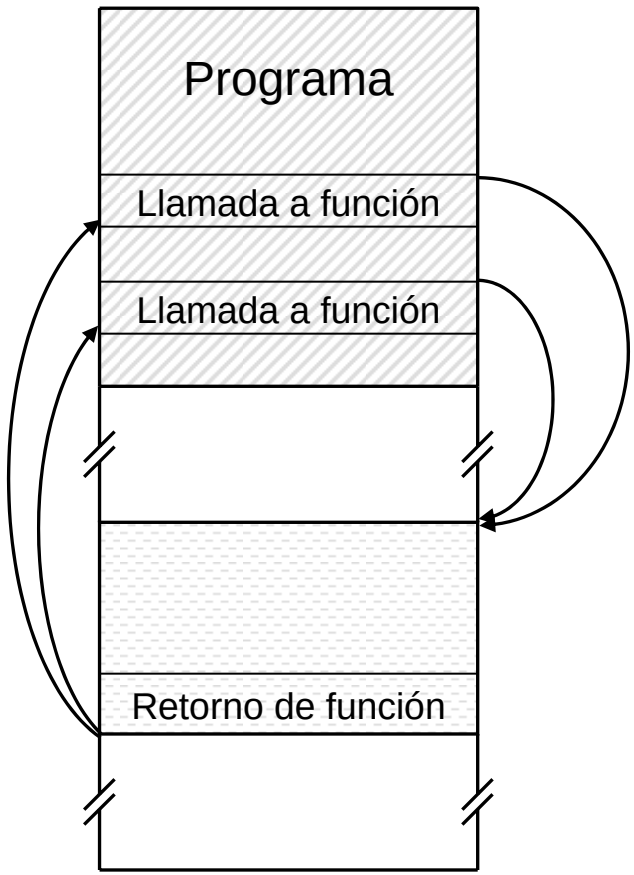
(también llamados funciones o procedimientos)

Programa en lenguaje de alto nivel (C)

```
{
  …
  res = Producto(5,3);
  …
}

int Producto(int a, int b)
{
  return a * b;
}
```

Parámetros o argumentos



En Simplez:

- Transmisión de argumentos por valor.
- Dirección de retorno construyendo **`BR #dir`**

Subprograma para restar

```
Dir   C. Hex   Cont. Nem.    Comentarios
0     H'60A    BR   #10     ; Salto prog,
1     H'610    BR   #16     ; Ret. llamada 1
2     H'61A    BR   #26     ; Ret. llamada 2
               …
10    H'201    LD   /1      ; Coloca instru
11    H'0D1    ST   /209    ; de retorno 1.
12    H'233    LD   /51     ; Transmite
13    H'0D3    ST   /211    ; sustraendo 1.
14    H'232    LD   /50     ; Tr. minuendo 1
15    H'6C8    BR   #200    ; Llama a Resta
16    H'034    ST   /52     ; Guarda result.
               …
20    H'202    LD   /2      ; Coloca instru
21    H'0D1    ST   /209    ; de retorno 2
22    H'265    LD   /101    ; Transmite
23    H'0D3    ST   /211    ; sustraendo 2
24    H'064    LD   /100    ; Tr. minuendo 2
25    H'6C8    BR   #200    ; Llama a Resta
26    H'066    ST   /102    ; Guarda result.
```

```
Dir   C. Hex   Cont. Nem.    Comentarios
; Función int Resta(int m, int s)
200   H'C00    DEC          ; Resta unidad
201   H'0D2    ST   /210    ; al minuendo.
202   H'2D3    LD   /211    ; El sustraendo
203   H'C00    DEC          ; es el contador
204   H'8D0    BZ   #208    ; Si cero, salir
205   H'0D3    ST   /211    ; Guarda sustra.
206   H'2D2    LD   /210    ; (minuendo)→ AC
207   H'6C8    BR   #200    ; Continua bucle
208   H'2D2    LD   /210    ; Valor retorno
209   H'600    BR   #0      ; Sustituir
```

MP[209] cuando **PC = 15**

BR #16

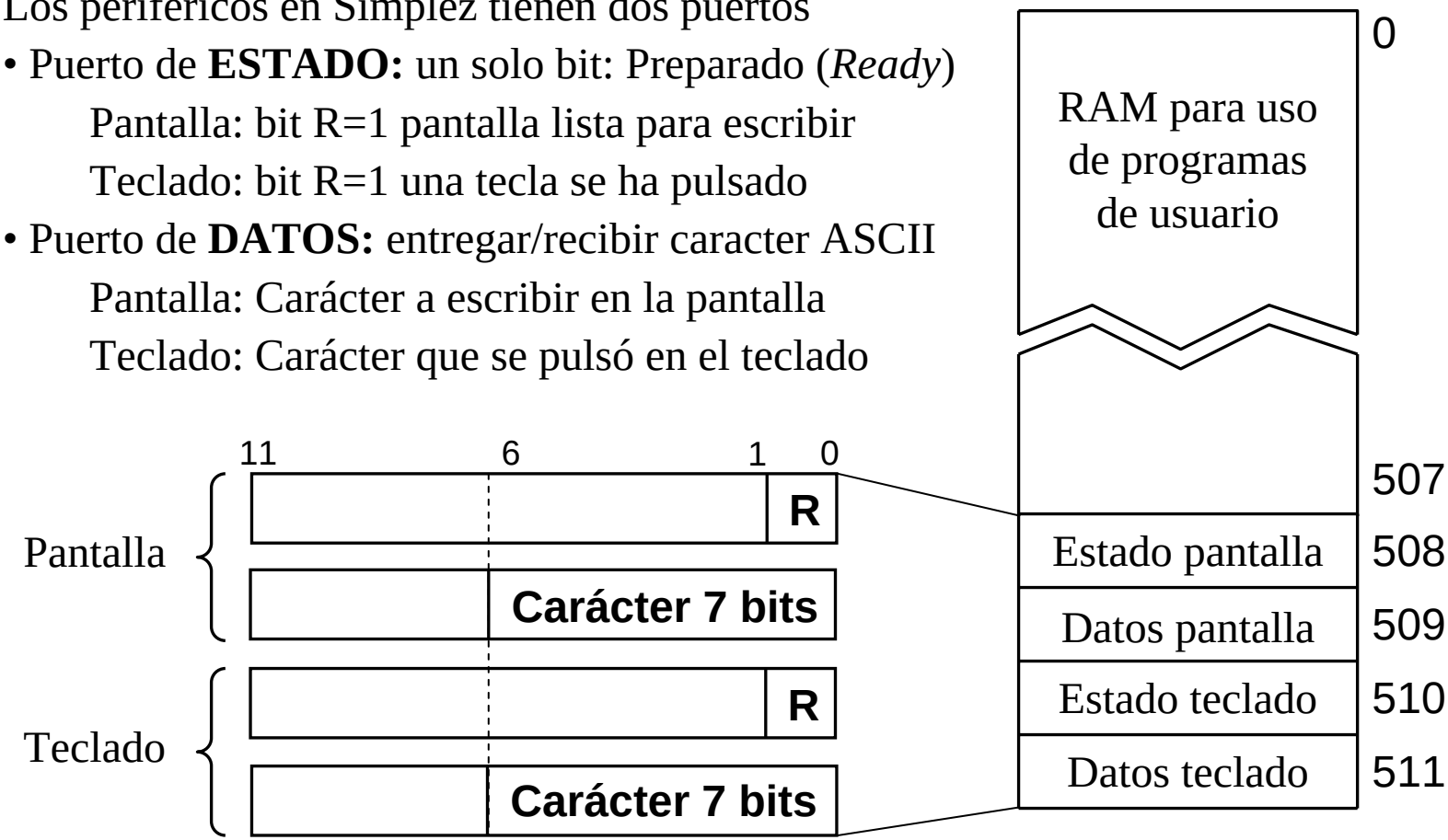
MP[209] cuando **PC = 25**

BR #26

Comunicación con los periféricos

Los periféricos en Simplez tienen dos puertos

- Puerto de **ESTADO:** un solo bit: Preparado (*Ready*)
  - Pantalla: bit R=1 pantalla lista para escribir
  - Teclado: bit R=1 una tecla se ha pulsado
- Puerto de **DATOS:** entregar/recibir caracter ASCII
  - Pantalla: Carácter a escribir en la pantalla
  - Teclado: Carácter que se pulsó en el teclado



Mecanismo de espera activa ("*busy waiting*")

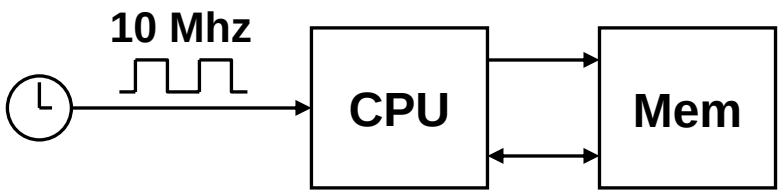
Para escribir en pantalla un carácter almacenado en la dirección 5 de la memoria:

```
Dir   C. Hex   Cont. Nem.    Comentarios
                …
15    H'3FC    LD  /508    ; Mientras estado
16    H'80F    BZ  #15     ; pantalla sea 0 espera...
17    H'205    LD  /5      ; Ya podemos escribir
18    H'1FD    ST  /509    ; un carácter en pantalla
                …
```

Para leer un carácter del teclado y almacenarlo en la dirección 6 de la memoria:

```
Dir   C. Hex   Cont. Nem.    Comentarios
                …
20    H'3FE    LD  /510    ; Mientras estado teclado
21    H'814    BZ  #20     ; sea 0 espera pulsación...
22    H'3FF    LD  /511    ; Una tecla se ha pulsado
23    H'006    ST  /6      ; Almacenamos el resultado
                …
```

Potencia computacional empleada en la espera activa



En Simplez:
- El reloj del sistema es de 10 MHz
- Cada ciclo de reloj tarda 100 ns
- La pantalla de Simplez puede imprimir 30 caracteres por segundo.

Tiempos de ejecución de las instrucciones en Simplez:
- ST: 3 ciclos = 300 ns
- LD: 3 ciclos = 300 ns
- ADD: 4 ciclos = 400 ns
- BR: 3 ciclos = 300 ns
- BZ: 3 ciclos = 300 ns
- CLR: 3 ciclos = 300 ns
- DEC: 3 ciclos = 300 ns
- HALT: detiene procesador

Escritura de un carácter en pantalla:

```
Dir   C. Hex   Cont. Nem.
                  …
15    H'3FC    LD  /508
16    H'80F    BZ  #15

17    H'205    LD  /5
18    H'1FD    ST  /509
                  …
```

Durante 1/30 de segundo, se ejecutan en este bucle:

$$n = 1/30\,s \cdot \frac{1\ \text{instrucción}}{300\,ns} = 111111\ \text{instrucciones}$$

Conclusiones

Simplez es un procesador muy útil pero ...

- Tiene un juego de instrucciones demasiado reducido (por ejemplo falta SUB).
- Las instrucciones aritméticas necesitan un acceso a memoria para leer el operando.
- Posee un único tipo de datos para realizar operaciones.
- Es necesario que el programa se automodifique para recorrer zonas de memoria.
- Carece de mecanismos sencillos para poder realizar llamadas a funciones.
- El espacio de direccionamiento del procesador es muy limitado.
- La gestión de los periféricos es ineficiente.

Solución: mejorar la estructura de la CPU (Simplez+i4):

- Instrucciones de restar y cargar valores inmediatos en el acumulador.
- Direccionamiento indexado, direccionamiento indirecto.
- Mecanismo de gestión de periféricos por interrupción.