Tema 4: Modelo de la máquina elemental Simplez-I^3

Dpto. Ingeniería de Sistemas y Automática
Escuela Superior de Ingenieros
Universidad de Sevilla

Índice

Bibliografía

- REF: Conceptos básicos de arquitectura y sistemas operativos
- AUTOR: Gregorio Fernández.
- PÁGs: Capítulo 3

Índice

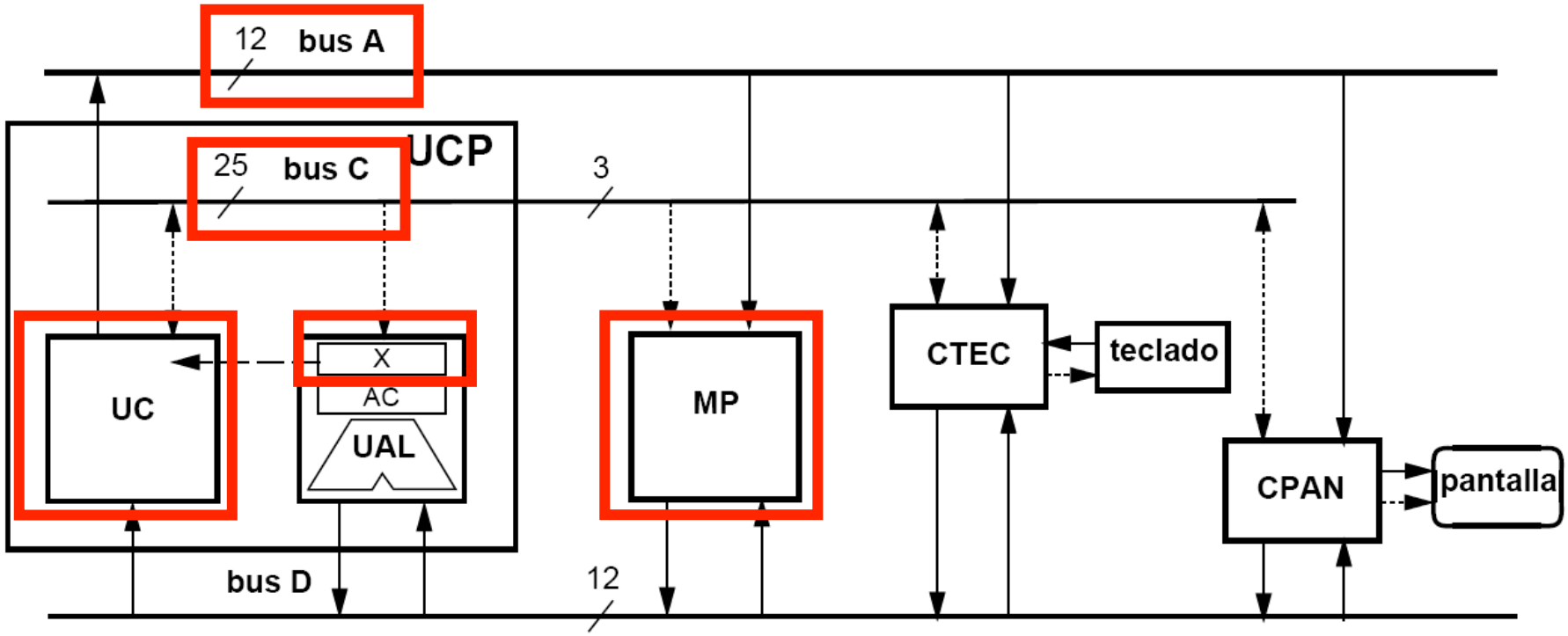
Bibliografía

- REF: Conceptos básicos de arquitectura y sistemas operativos
- AUTOR: Gregorio Fernández.
- PÁGs: Capítulo 3

Modelo estructural

- Diferencias con Símplez:
  - Se añade un registro de 12 bits en la UCP, al que llamaremos registro X (registro de índice).
  - Bus A pasa a tener 12 bits. MP puede tener ahora 4096 palabras (las cuatro últimas, reservadas para la entrada y la salida):
    - 4092: Estado pantalla
    - 4093: Dato pantalla
    - 4094: Estado teclado
    - 4095: Dato teclado
  - Bit Z:
    - Z=1 si el resultado de la última operación en la UAL ha sido 0.

Modelo estructural



Índice

Bibliografía

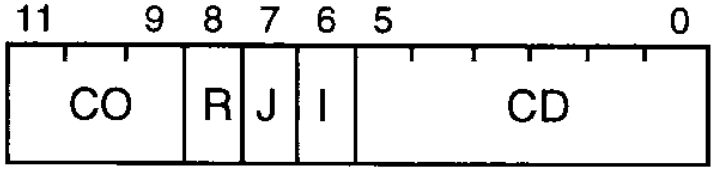
- REF: Conceptos básicos de arquitectura y sistemas operativos
- AUTOR: Gregorio Fernández.
- PÁGs: Capítulo 3

Modos de direccionamiento

- Concepto de dirección efectiva: Dirección de la MP donde está el dato.
- Criterio de notación: Utilizar paréntesis para indicar contenido, sea de un registro o de un campo de un formato o de una posición de memoria.
  - (AC) representará el contenido del acumulador
  - (CD) el contenido del campo CD del formato de instrucciones
  - (4) el contenido de la posición de memoria 4 (equivalente a MP[4])
- Símplez:
  - El direccionamiento es únicamente directo: lo que contenga el campo CD es la dirección de referencia en la MP, o dirección efectiva, DE.
    - LD /4 (la dirección efectiva del dato que quiero cargar en el acumulador es 4): AC ← (4) ó AC ← MP[4]

Modos de direccionamiento

- Formato de las instrucciones en Símplez+I[3]:



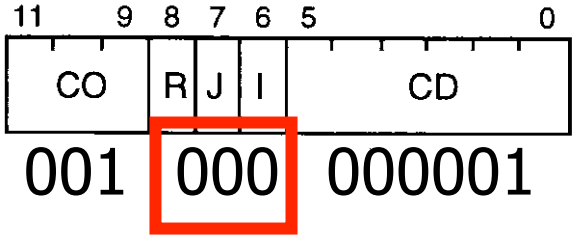
  - CD sólo tiene seis bits, por lo que directamente sólo pueden direccionarse 2^6 = 64 palabras de la MP.
  - El bit 8 ("R") indica el registro (AC ó X) con el que se opera (si la instrucción no opera con registro, no importa el valor de R):
    - R=1: se opera con registro X
    - R=0: se opera con registro AC
  - Los bits 7 y 6 ("J" e "I") indican el modo de direccionamiento:
    - J=0, I=0: direccionamiento directo (igual que en Símplez)
    - J=0, I=1: direccionamiento indirecto
    - J=1, I=0: direccionamiento indexado
    - J=1, I=1: direccionamiento indirecto e indexado

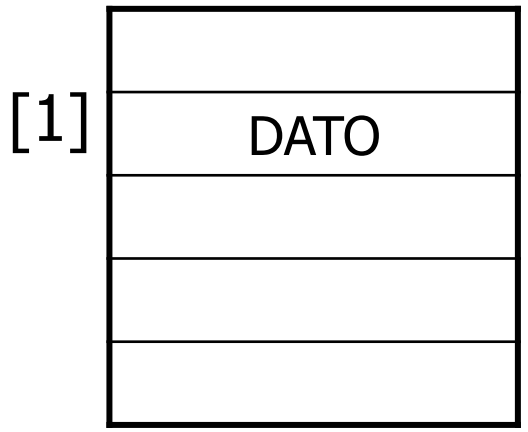
Modos de direccionamiento

- Direccionmaiento directo:
  - Lo que contenga el campo CD es, directamente, la dirección de referencia en la MP, o dirección efectiva, DE.

$$DE=(CD)$$

LD 1 (directo): LD /1





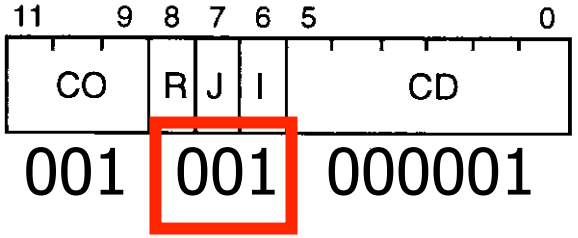
DE=(CD)=1

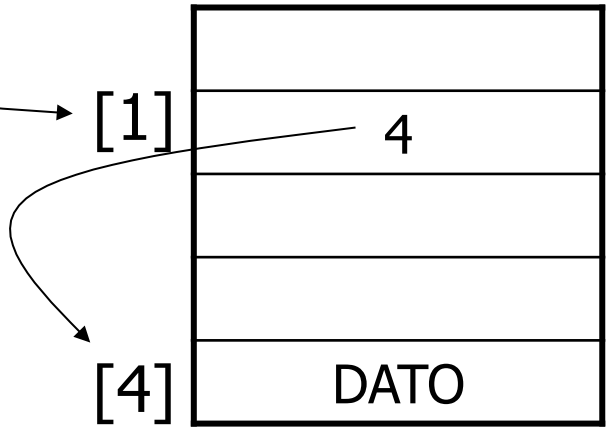
Modos de direccionamiento

- Direccionamiento indirecto:
  - Lo que contenga el campo CD es la dirección de una palabra de la MP que contiene la dirección efectiva (es decir, CD tiene la dirección del puntero al operando).

$$DE=((CD))$$

LD 1 indirecto: LD [/1]





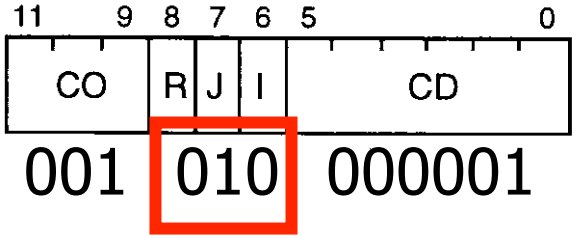
DE=((CD))=(1)=4

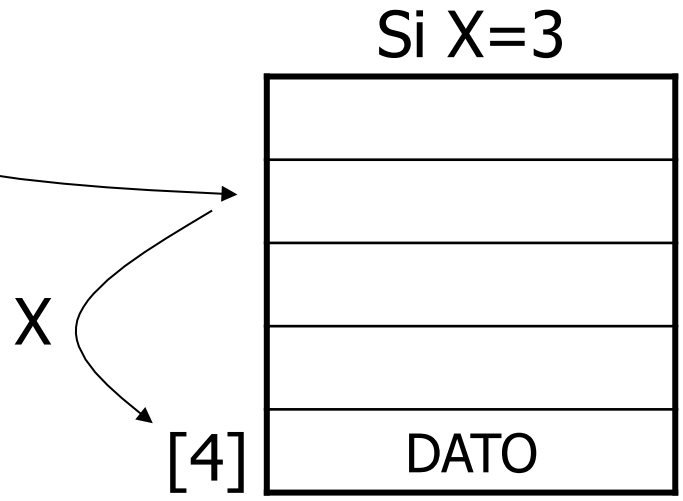
Modos de direccionamiento

- Direccionamiento indexado:
  - Para calcular la dirección efectiva hay que sumar el contenido del registro de índice, X.

$$DE=(CD) + (X)$$

LD 1 indexado: LD /1[.X]





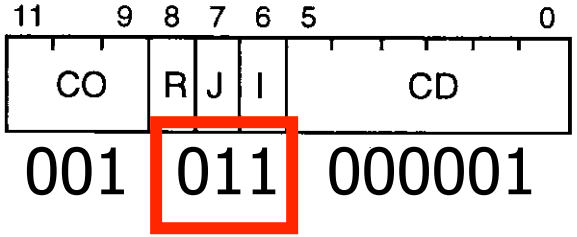
DE=(CD) + (X)= 1 + 3 = 4

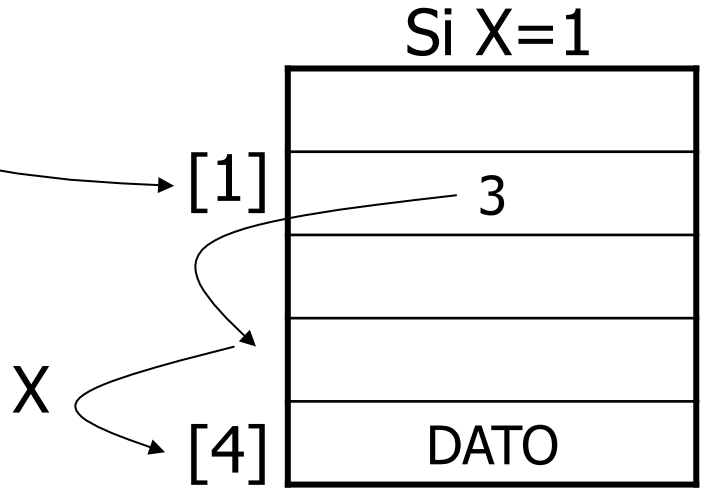
Modos de direccionamiento

- Direccionamiento indirecto e indexado:
  - Aplicación conjunta de los dos tipos de direccionamiento. El convenio de orden es postindexación, es decir, primero se acude al puntero para buscar la dirección (direccionamiento indirecto) y luego se suma el contenido de X (direccionamiento indexado).

DE=((CD))+(X) ~~DE=((CD)+(X))~~

LD 1 indirecto e indexado: LD [/1][.X]





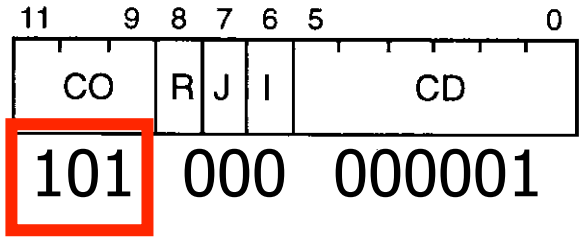
DE=((CD)) + (X)= (1)+(X)= 3+(X) = 3+1 = 4

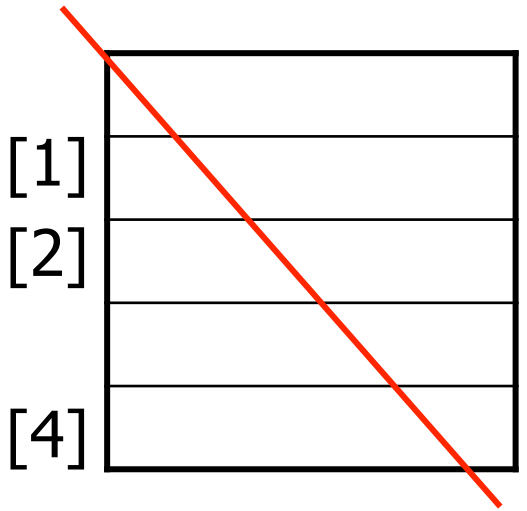
Modos de direccionamiento

- Direccionamiento inmediato:
  - Tenemos un operando inmediato.
    - El operando es directamente el contenido del CD
  - Se habla de "direccionamiento inmediato", pero no se direcciona la MP, ya que el operando se extrae de la misma instrucción.

Operando = (CD)

LD 1 inmediato: LD #1





~~DE~~ Operando = DATO = (CD) = 1

Modos de direccionamiento

- Direccionamiento inmediato:
  - La utilidad de este modo de direccionamiento se encuentra cuando se quiere introducir un valor inicial en el registro AC o en el registro X: evita la necesidad de tener previamente almacenado ese valor en la MP y leerlo de ésta.
  - El valor ha de ser una constante comprendida entre 0 y 63 (que es lo que "da de sí" el campo CD).
  - También se utiliza con la instrucción de resta.

Modos de direccionamiento

- Tabla resumen:

JI	modo	DE
00	directo	DE=(CD)
01	indirecto	DE=((CD))
10	indexado	DE=(CD)+(X)
11	indirecto e indexado	DE=((CD))+(X)
$\phi\phi$	inmediato	operando=(CD)

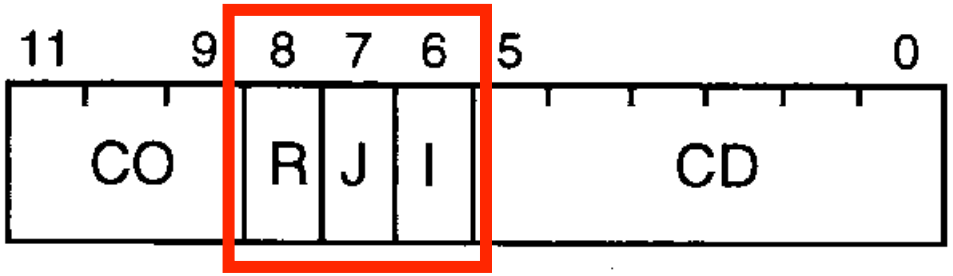
Índice

Bibliografía

- REF: Conceptos básicos de arquitectura y sistemas operativos
- AUTOR: Gregorio Fernández.
- PÁGs: Capítulo 3

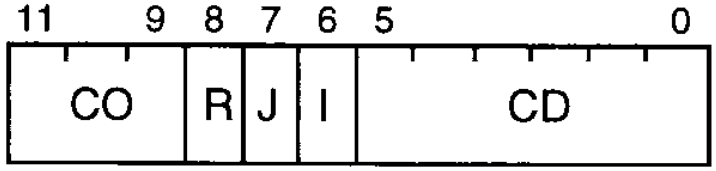
Modelo funcional

- Modelo funcional
  - Convenio representación datos e instrucciones
  - Repertorio de instrucciones
- Convenio de representación de datos e instrucciones:
  - Datos (igual que en Símplez)
    - Números enteros no negativos
    - Caracteres
  - Formato de instrucciones:



Modelo funcional

- Formato instrucciones:



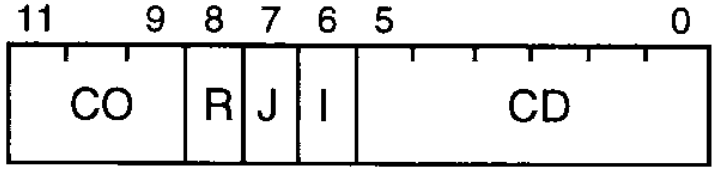
  - CD sólo tiene seis bits, por lo que directamente sólo pueden direccionarse 2^6 = 64 palabras de la MP.
  - El bit 8 ("R") indica el registro (AC ó X) con el que se opera (si la instrucción no opera con registro, no importa el valor de R):
    - R=1: se opera con registro X
    - R=0: se opera con registro A
  - Los bits 7 y 6 ("J" e "I") indican el modo de direccionamiento:
    - J=0, I=0: direccionamiento directo (igual que en Símplez)
    - J=0, I=1: direccionamiento indirecto
    - J=1, I=0: direccionamiento indexado
    - J=1, I=1: direccionamiento indirecto e indexado

Modelo funcional

- Repertorio de instrucciones:
  - Las instrucciones "CLR" y "DEC" (101 y 110) quedarán sustituidas por otras más potentes: LD# y SUB#.
  - Las demás instrucciones tienen los mismos significados que en Símplez, con las siguientes matizaciones:
    - "ST", "LD" y "ADD" (000, 001 y 010) pueden almacenar, cargar, o sumar a, respectivamente, el acumulador AC o el registro X.
    - La condición para la bifurcación en "BZ" sigue siendo que el resultado de la última operación en la UAL haya sido cero. En Símplez esto es equivalente a decir que el contenido del acumulador sea cero, pero NO EN SÍMPLEZ+I[3], porque los resultados pueden ir al acumulador o al registro X.
    - Instrucciones que pueden afectar al bit Z: LD, ADD, LD# y SUB#.
    - Solamente se puede restar en inmediato!!!

Modelo funcional

- Repertorio de instrucciones:



CO (bin.)	CO (oct.)	CO (nem.)	Significado
000	0	ST	(AC) → MP[DE], o (X) → MP[DE]
001	1	LD	(MP[DE]) → AC, o (MP[DE]) → X
010	2	ADD	(AC) + (MP[DE]) → AC, o (X) + (MP[DE]) → X
011	3	BR	siguiente instrucción en MP[DE]
100	4	BZ	si cero en UAL, sig. instr. en MP[DE]
101	5	LD #	(CD) → AC, o (CD) → X
110	6	SUB #	(AC) – (CD) → AC, o (X) – (CD) → X
11100	70, 71	HALT	detiene la ejecución

Modelo funcional

- Resumen: direcciones VS instrucciones:
  - Si el código de operación es 000, 001 ó 010 (ST, LD o ADD), entonces el modo de direccionamiento para llegar al operando viene dado por la tabla del apartado 2.
  - Si el código de operación es 011 ó 100 (BR o BZ) el valor que tenga el bit 8 es indiferente y el modo de direccionamiento para llegar a la dirección de bifurcación viene dado también con la tabla del apartado 2.
  - Si el código de operación es 101 (LD # cargar literal) ó 110 (SUB # restar literal) el modo de direccionamiento es siempre inmediato.

Índice

Bibliografía

- REF: Conceptos básicos de arquitectura y sistemas operativos
- AUTOR: Gregorio Fernández.
- PÁGs: Capítulo 3

Convenios simbólicos

- Seguiremos utilizando los convenios simbólicos de Símplez (códigos de operación nemónicos, direcciones en decimal).

- Pero ahora tenemos que ampliar esos convenios para incluir la información sobre:
  - El registro utilizado
  - El modo de direccionamiento

Convenios simbólicos

- Registro utilizado:
  - Escribiremos, a continuación del código, ".A" o ".X" para indicar que se opera con AC o con X
  - Después, si la instrucción hace referencia a MP, una coma y finalmente la dirección.

binario	octal	simbólico
000000000010	0002	ST .A,/2
000100000010	0402	ST .X,/2
001100000111	1407	LD .X,/7

"Cargar en el registro X el contenido de la palabra cuya dirección es 7".

Convenios simbólicos

- Modos de direccionamiento:
  - Direccionamiento directo

- *Ejemplos de instrucciones con direccionamiento directo:*

Binario	Octal	Simbólico	Efecto
010000100000	2040	`ADD .A,/32`	(AC) + (32) → AC
010100100000	2440	`ADD .X,/32`	(X) + (32) → X
100000000101	4005	`BZ /5`	Si el último resultado en la UAL fue 0, bifurca a la dirección 5

Convenios simbólicos

- Modos de direccionamiento:
  - Direccionamiento indirecto: Se indica el direccionamiento indirecto mediante corchetes.

binario	octal	simbólico
000001000010	0102	ST .A,[/2]
000101000010	0502	ST .X,[/2]
010101000011	2503	ADD .X,[/3]

"Sumar al contenido de X el contenido de la palabra cuya dirección efectiva es MP[3], dejando el resultado en X".

Convenios simbólicos

- Modos de direccionamiento:
  - Direccionamiento indirecto

- *Ejemplos de direccionamiento indirecto:*

Binario	Octal	Simbólico	Efecto
010001100000	2140	`ADD .A,[/32]`	(AC) + ((32)) → AC
010101100000	2540	`ADD .X,[/32]`	(X) + ((32)) → X
100001000101	4105	`BZ [/5]`	Si el último resultado en la UAL fue 0, bifurca a la dirección contenida en la palabra de dirección 5

Convenios simbólicos

- Modos de direccionamiento:
  - Direccionamiento indexado: Se indica que escribiendo [.X] tras la dirección.

binario	octal	simbólico
000010000010	0202	ST .A,/2[.X]
010010000011	2203	ADD .A,/3[.X]
000110000010	0602	ST .X,/2[.X]
010110000011	2603	ADD .X,/3[.X]

"Sumar al contenido de X el contenido de la palabra cuya dirección efectiva es 3+(X), y dejar el resultado en X".

Convenios simbólicos

- Modos de direccionamiento:
  - Direcionamiento indexado

- *Ejemplos de direccionamiento indexado:*

Binario	Octal	Simbólico	Efecto
010010100000	2240	`ADD .A,/32[.X]`	(AC) + (32 + (X)) → AC
010110100000	2640	`ADD .X,/32[.X]`	(X) + (32 + (X)) → X (raro, pero válido)
100010000101	4205	`BZ /5[.X]`	Si el último resultado en la UAL fue 0, bifurca a la dirección 5 + (X)

Convenios simbólicos

- Modos de direccionamiento:
  - Direccionamiento indirecto e indexado: Se aplican a la vez los 2 criterios indicados para el direccionamiento indirecto e indexado.

binario	octal	simbólico
000011000010	0302	ST .A,[/2][.X]
010011000011	2303	ADD .A,[/3][.X]
000111000010	0702	ST .X,[/2][.X]
010111000011	2703	ADD .X,[/3][.X]

"Sumar al contenido de X el contenido de la palabra cuya dirección efectiva es MP[3]+(X), y dejar el resultado en X".

Convenios simbólicos

- Modos de direccionamiento:
  - Direccionamiento indirecto e indexado

- *Ejemplos de direccionamiento indirecto–indexado:*

Binario	Octal	Simbólico	Efecto
010011100000	2340	`ADD .A,[/32][.X]`	(AC) + ((32) + (X)) → AC
010111100000	2740	`ADD .X,[/32][.X]`	(X) + ((32) + (X)) → X (raro, pero válido)
100011000101	4305	`BZ [/5][.X]`	Si último resultado en la UAL 0, bifurca a la dirección (5) + (X)

Convenios simbólicos

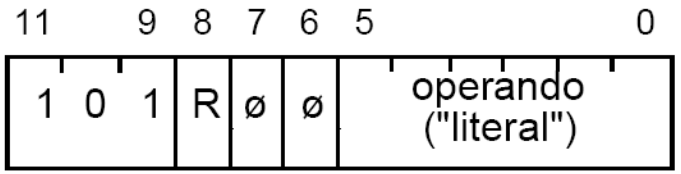
- Modos de direccionamiento:
  - Direccionamiento inmediato: Se representa con # delante del operando inmediato. Para cargar utilizamos el mismo nemónico de la carga con direccionamiento (LD) y para la resta SUB.

binario	octal	simbólico
101000000010	5002	LD .A, #2
101011000010	5302	LD .A, #2
101100001010	5412	LD .X, #10
101111001010	5712	LD .X, #10
110000000001	6001	SUB .A, #1
110010000001	6201	SUB .A, #1

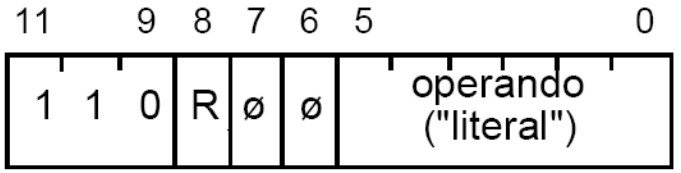
"Cargar un 10 en el registro X" .

Convenios simbólicos

- Modos de direccionamiento:
  - Direccionamiento inmediato



(LD #)



(SUB #)

Binario	Octal	Simbólico	Efecto
101000111111	5077	LD .A,#63	63 → AC
101010111111	5277	LD .A,#63	63 → AC
101100111111	5477	LD .X,#63	63 → X
101111111111	5777	LD .X,#63	63 → X
110000000001	6001	SUB .A,#1	(AC) – 1 → AC
110100101000	6450	SUB .X,#40	(X) – 40 → X

Índice

Bibliografía

- REF: Conceptos básicos de arquitectura y sistemas operativos
- AUTOR: Gregorio Fernández.
- PÁGs: Capítulo 3

Ejemplo 1: Sucesión de Fibonacci

- Sumar los 10 primeros términos de la sucesión

  - Pseudocódigo:

```
PEN=0; ULT=1;
SUM=1; CONT=8;
Mientras CONT≠0
        SIG=PEN+ULT;
        SUM=SUM+SIG;
        PEN=ULT;
        ULT=SIG;
        CONT=CONT-1;
FinMientras
```

Ejemplo 1: Sucesión de Fibonacci

```
        ORG 0
        BR /INI
PEN     RES 1
ULT     RES 1
SIG     RES 1
SUM     RES 1
INI     LD .A, #0
        ST .A, /PEN
        LD .A, #1
        ST .A, /ULT
        ST .A, /SUM
        LD .X, #8
BUCLE   BZ /FIN
        LD .A, /PEN
        ADD .A, /ULT
        ST .A, /SIG
        ADD .A, /SUM
        ST .A, /SUM
        LD .A, /ULT
        ST .A, /PEN
        LD .A, /SIG
        ST .A, /ULT
        SUB .X, #1
        BR /BUCLE
FIN     HALT
```

Ejemplo 2: Suma 100 elementos de una tabla

- Sumar los 100 números situados entre las posiciones 50 y 149, dejando el resultado en la posición 150.

- 1) Versión indexada. Pseudocódigo:

```
PSUMA = 150;
*PSUMA = 0;
X = 100;
Mientras X≠0
    X = X - 1;
    *PSUMA = *PSUMA + TAB[X];
FinMientras
```

Ejemplo 2: Suma 100 elementos de una tabla

Ensamblador

```
        ORG 0
        BR /INIPRG
PSUMA   DATA SUMA
CIEN    DATA 100
INIPRG  LD .A, #0
        ST .A, [/PSUMA]
        LD .X, /CIEN
BUCLE   BZ /FIN
        SUB .X, #1
        ADD .A, /TAB[.X]
        SUB .X, #0
        BR /BUCLE
FIN     ST .A, [/PSUMA]
        HALT
        ORG 50
TAB     DATA 23
        .
        .
        .
        DATA 31
SUMA    RES 1
        END
```

Ejemplo 2: Suma 100 elementos de una tabla

- Sumar los 100 números situados entre las posiciones 50 y 149, dejando el resultado en la posición 150.

- 2) Versión con punteros. Pseudocódigo:

```
PSUMA=150; PTAB=50;
*PSUMA=0;
CONT=100;
Mientras CONT≠0
    *PSUMA = *PSUMA + *PTAB;
    PTAB = PTAB + 1 ;
    CONT = CONT -1;
FinMientras
```

Ejemplo 2: Suma 100 elementos de una tabla

```
        ORG 0
        BR /INIPRG
PSUMA   DATA SUMA
DTAB    DATA TAB
PTAB    RES 1
CONT    RES 1
CIEN    DATA 100
INIPRG  LD .A, #0
        ST .A, [/PSUMA]
        LD .A, /DTAB
        ST .A, /PTAB
        LD .A, /CIEN
        ST.A, /CONT
BUCLE   BZ /FIN
        LD .A, [/PTAB]
        ADD .A, [/PSUMA]
        ST .A, [/PSUMA]
        LD .A, #1
        ADD.A, /PTAB
        ST .A, /PTAB
        LD.A, /CONT
        SUB .A, #1
        ST .A, /CONT
        BR /BUCLE
FIN     HALT
        ORG 50
TAB     DATA 23
        .
        .
        .
        DATA 31
SUMA    RES 1
        END
```

Ejemplo 2: Suma 100 elementos de una tabla

- Sumar los 100 números situados entre las posiciones 50 y 149, dejando el resultado en la posición 150.

- 2) Versión con punteros. Pseudocódigo:

```
PSUMA=150; PTAB=149;
*PSUMA=0;
X=100;
Mientras X≠0
    *PSUMA = *PSUMA + *PTAB;
    PTAB = PTAB -1 ;
    X = X -1;
FinMientras
```

Ejemplo 2: Suma 100 elementos de una tabla

```
        ORG 0
        BR /INIPRG
PSUMA   DATA SUMA
DTAB    DATA FTAB
PTAB    RES 1
CIEN    DATA 100
INIPRG  LD .A, #0
        ST .A, [/PSUMA]
        LD .A, /DTAB
        ST .A, /PTAB
        LD .X, /CIEN
```

```
BUCLE   BZ /FIN
        LD .A, [/PTAB]
        ADD .A, [/PSUMA]
        ST .A, [/PSUMA]
        LD .A, /PTAB
        SUB .A, #1
        ST .A, /PTAB
        SUB .X, #1
        BR /BUCLE
FIN     HALT
```

```
        ORG 50
TAB     DATA 23
        .
        .
        .
FTAB    DATA 31
SUMA    RES 1
        END
```

Ejemplo 3: Uso de subprogramas (resta)

- Realizar resta de dos valores utilizando una rutina:
  - Paso de minuendo y sustraendo por las direcciones fijas de etiqueta MINU y SUSTR respectivamente.
  - El resultado se devuelve por dirección fija a través de la etiqueta RESULT.
- Pseudocódigo:

```
<Principal>
    ...
    E=Resta(B,A);
    ...
    F=Resta(D,C);
    ...
<Fin>
```

```
<Resta> (MINU,SUSTR)
    RESULT=MINU;
    Mientras SUSTR≠0
        RESULT--;
        SUSTR--;
    FinMientras
    Devuelve RESULT;
<Fin>
```

Ejemplo 3: Uso de subprogramas (resta)

Principal (direcciones de retorno entre 0 y 63)

```
        ...
        LD .A, /B
        ST .A, /MINU
        LD .A, /A
        ST .A, /SUSTR
        LD .A, #DRET1
        ST .A, /DIRRET
        BR /RESTA
DRET1   LD .A, /RESULT
        ST .A, /E
        ...
```

```
        ...
        LD .A, /D
        ST .A, /MINU
        LD .A, /C
        ST .A, /SUSTR
        LD .A, #DRET2
        ST .A, /DIRRET
        BR /RESTA
DRET2   LD .A, /RESULT
        ST .A, /F
        ...
```

Ejemplo 3: Uso de subprogramas (resta)

Principal (direcciones de retorno > 63)

```
                ORG 0
                BR /INIPRG
DDRET1          DATA DRET1
                ...
                ORG 62
INIPRG          LD .A, /B
                ST .A, /MINU
                LD .A, /A
                ST .A, /SUSTR
                LD .A, /DDRET1
                ST .A, /DIRRET
                BR /RESTA
DRET1           LD .A, /RESULT
                ST .A, /E
                ...
```

Ejemplo 3: Uso de subprogramas (resta)

Subprograma

```
RESTA     BR /INIRUT
MINU      RES 1
SUSTR     RES 1
RESULT    RES 1
DIRRET    RES 1
INIRUT    LD .A, /MINU
          LD .X, /SUSTR
BUCLE     BZ /FINRESTA
          SUB .A, #1
          SUB .X, #1
          BR /BUCLE
FINRESTA  ST .A, /RESULT
          BR [/DIRRET]
```

Ejemplo 3: Uso de subprogramas (resta)

Subprograma (salvar y restaurar registros!)

```
RESTA     BR /INIRUT
MINU      RES 1
SUSTR     RES 1
RESULT    RES 1
DIRRET    RES 1
RA        RES 1
RX        RES 1
INIRUT    ST .A, /RA
          ST .X, /RX
          LD .A, /MINU
          LD .X, /SUSTR
BUCLE     BZ /FINRESTA
          SUB .A, #1
          SUB .X, #1
          BR /BUCLE
FINRESTA  ST .A, /RESULT
          LD .A, /RA
          LD .X, /RX
          BR [/DIRRET]
```

Ejemplo 3: Uso de subprogramas (resta)

Subprograma (otro modo de pasar los parámetros)

```
PRUT    DATA RESTA
        ...
        LD .X, #1
        LD .A, /B
        ST .A, [/PRUT][.X]
        LD .X, #2
        LD .A, /A
        ST .A, [/PRUT][.X]
        LD .X, #4
        LD .A, #DRET1
        ST .A, [/PRUT][.X]
        BR [/PRUT]
DRET1   LD .X, #3
        LD .A, [/PRUT][.X]
        ST .A, /E
```

```
RESTA   BR [/PINIRUT]
        (parámetro ent. 1)
        (parámetro ent. 2)
        (parám. salida)
        (dir. retorno)
        ...
```

Ejemplo 4: Problema de inicializar con DATA

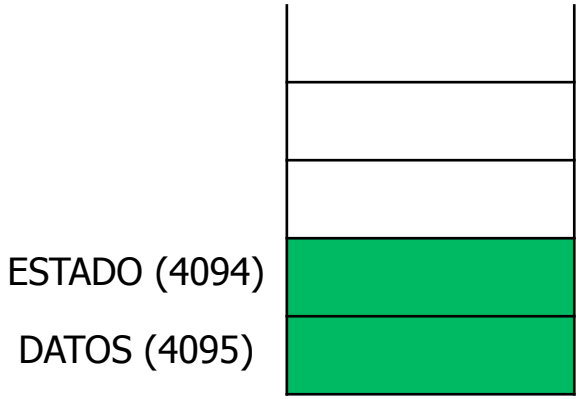
Ejecuciones sucesivas deben funcionar sin recargar!

```
        BR /INI
VALOR   DATA 100
VAR     RES 1
CONT    DATA 20
INI     LD .A, /VALOR
        ST .A, /VAR
```

```
BUCLE   LD .A, /CONT
        BZ /FIN
        SUB .A, #1
        ST .A, /CONT
        LD .A, /VAR
        SUB .A, #1
        ST .A, /VAR
        BR /BUCLE
FIN     HALT
```

Ejemplo 5: Llenar vector por teclado

- En la memoria de un ordenador Símplez+i[3] se pretende almacenar un vector. El vector comienza en una dirección cuya etiqueta es DV, y cada uno de los elementos ocupa una palabra de memoria.
- Se pide realizar un programa en ensamblador que introduzca datos en el vector, leyéndolos del teclado por espera activa. Para realizar esto hay que tener en cuenta que por el puerto de entrada aparecerán sucesivamente el índice que indica la posición que ocupa el elemento en el vector y el valor del elemento. La última posición a la que se da valor es la posición 0. El programa parará después de almacenar el valor correspondiente.
- Las direcciones del puerto de entrada tienen como etiquetas ESTADO y DATOS, y se suponen definidas.
- La etiqueta DV se supone definida.
- El programa deberá funcionar independientemente de la posición a que apunten DV, ESTADO y DATOS.
- Se supone que los índices y valores de los elementos del vector son inferiores a 10.



Ejemplo 5: Llenar vector por teclado

- Pseudocódigo:

```
X=0;
PV=DV;
Hacer
    Mientras ESTADO==0
    Fin Mientras
    X = DATOS;
    Mientras ESTADO==0
    Fin Mientras
    VALOR=DATOS;
    PV[X]=VALOR;
Mientras X!=0
```

Ejemplo 5: Llenar vector por teclado

```
        ORG 0
        BR /INI
DDV     DATA DV
DDATOS  DATA DATOS
DESTADO DATA ESTADO
PV      RES 1
INI     LD .X, #0
        LD .A, /DDV
        ST .A, /PV
BUCLE
ESP1    LD .A, [/DESTADO]
        BZ /ESP1
        LD .X, [/DDATOS]
        SUB .X, #48
ESP2    LD .A, [/DESTADO]
        BZ /ESP2
        LD .A, [/DDATOS]
        SUB .A, #48
        ST .A, [/PV][.X]
        SUB .X, #0
        BZ /FIN
        BR /BUCLE
FIN     HALT
```

Ejemplo 5: Llenar vector por teclado

- Pseudocódigo:

```
Hacer
    INDICE = LeerTeclado();
    VALOR = LeerTeclado();
    DV[INDICE] = VALOR;
Mientras INDICE!=0
```

Ejemplo 5: Llenar vector por teclado

```
            ORG 0
            BR /BUCLE
DDV         DATA DV
DDATOS      DATA DATOS
DESTADO     DATA ESTADO
INDICE      RES 1
VALOR       RES 1
```

```
BUCLE
ESP1    LD .A, [/DESTADO]
        BZ /ESP1
        LD .A, [/DDATOS]
        SUB .A, #48
        ST .A, /INDICE
ESP2    LD .A, [/DESTADO]
        BZ /ESP2
        LD .A, [/DDATOS]
        SUB .A, #48
        ST .A, /VALOR
        LD.X, /INDICE
        LD.A, /VALOR
        ST .A, [/DDV][.X]
        SUB .X, #0
        BZ /FIN
        BR /BUCLE
FIN     HALT
```

Ejemplo 6: Subrutina para multiplicar

- Realizar una rutina en ensamblador de Símplez+i[3] que multiplica dos números.
- La rutina recibe en el registro AC una dirección de memoria. A partir de dicha dirección se encuentran almacenados (por este orden), el primer factor, el segundo factor y la dirección de retorno; el resultado se devuelve en el registro AC.
- Las instrucciones de la rutina se ensamblarán a partir de la dirección 20; no existe programa principal. Es preciso definir las etiquetas que se utilicen y explicar las operaciones realizadas.

Ejemplo 6: Subrutina para multiplicar

```
            ORG 20
MULTIP      BR /INIRUT
F1          RES 1
F2          RES 1
DIR         RES 1
DRET        RES 1
RX          RES 1
INIRUT      ST .X, /RX
            ST .A /DIR
            LD .A, [/DIR]
            ST .A, /F1
            LD .X, #1
            LD .A, [/DIR][.X]
            ST .A, /F2
            LD .X, #2
            LD .A, [/DIR][.X]
            ST .A, /DRET
            LD .A, #0
            LD .X, /F2
BUCLE       BZ /FRUT
            ADD .A, /F1
            SUB .X, #1
            BR /BUCLE
FRUT        LD .X, /RX
            BR [/DRET]
```

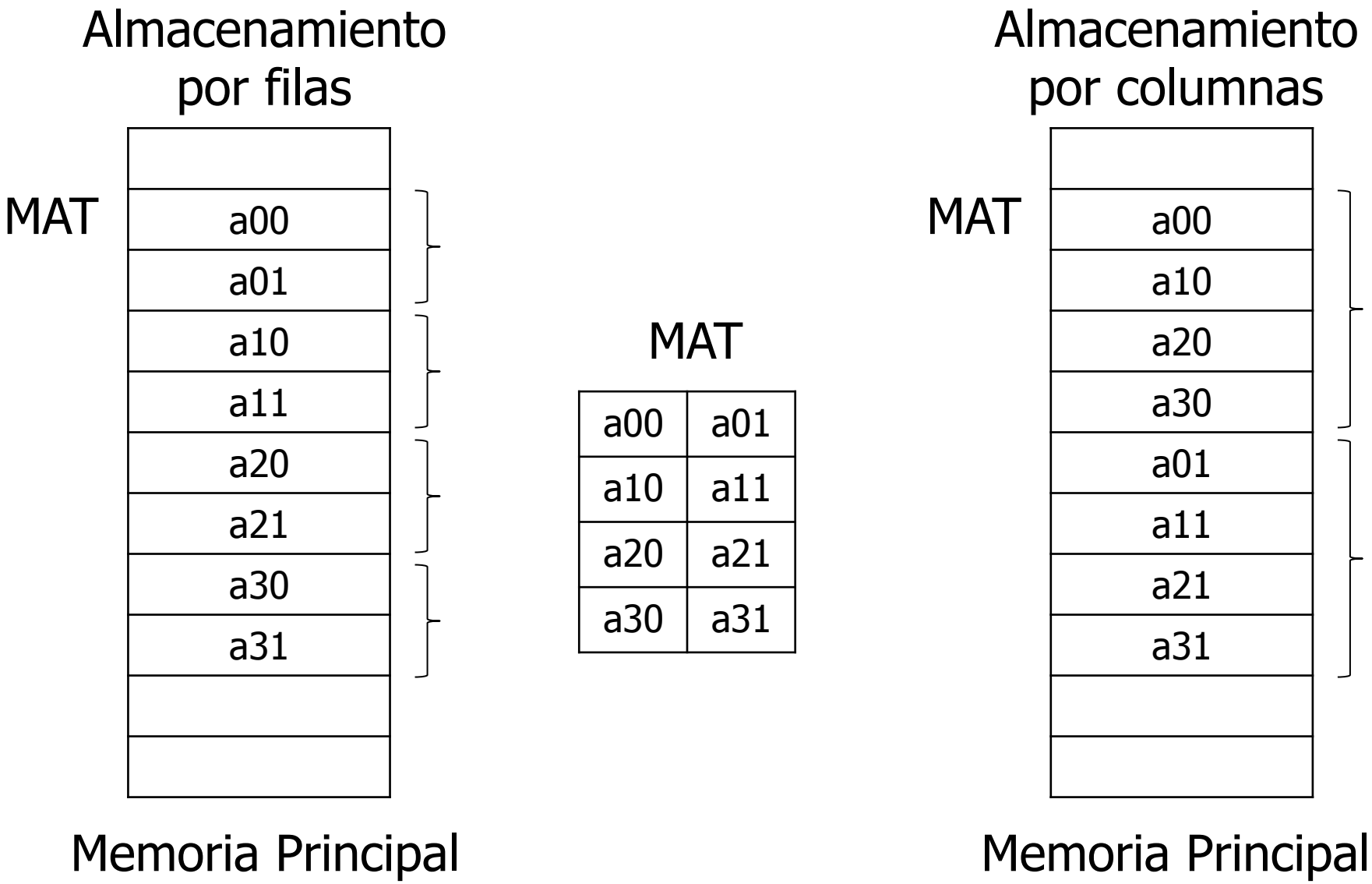
Ejemplo 7: Acceder a los elementos de una matriz

- En un computador Símplez-i^3 tenemos almacenada una matriz de datos en la zona alta de memoria a partir de la dirección de etiqueta MAT. El tamaño de la matriz está almacenado en las direcciones de etiqueta NFIL y NCOL en zona baja de memoria.
- La matriz se encuentra almacenada por filas en memoria.
- Se pide realizar un programa en Símplez-i^3 que imprima por pantalla 5 elementos de la matriz solicitados por el usuario a través del teclado.
- El programa leerá primero la fila y después la columna especificadas por el usuario e imprimirá por pantalla el elemento de la matriz correspondiente.
- Puede suponer que los índices dados por el usuario son siempre correctos, es decir están en los rangos [0,NFIL-1] y [0,NCOL-1]
- Los índices de la matriz y los valores guardados son menores que 10.

Ejemplo 7: Acceder a los elementos de una matriz

- Almacenamiento de matrices en memoria:
  - Las matrices se almacenan en memoria de forma lineal.
  - Los elementos de la matriz se pueden ordenar de diferentes maneras en memoria.
  - Las formas de almacenamiento más usuales son:
    - Almacenamiento por filas: La matriz se guarda en memoria de modo que se encuentran los datos de una fila a continuación de los de la anterior, empezando por la primera y acabando por la última.
    - Almacenamiento por columnas: La matriz se guarda en memoria de tal modo que se encuentran los datos de una columna a continuación de los de la anterior, empezando por la primera columna y acabando por la última.

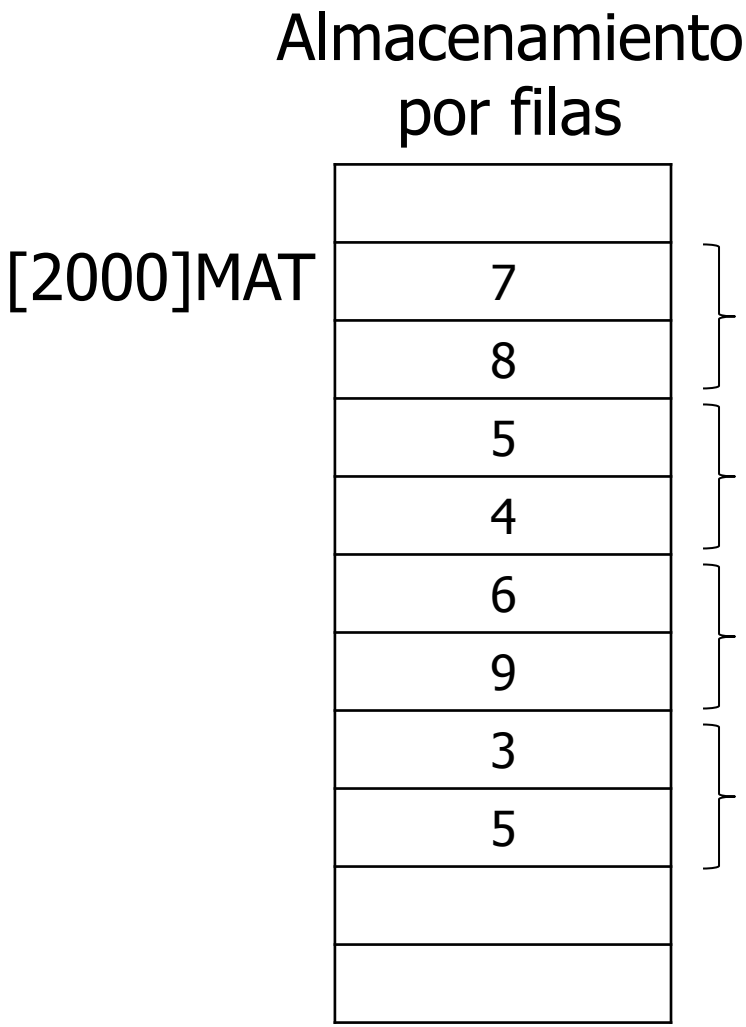
Ejemplo 7: Acceder a los elementos de una matriz



Ejemplo 7: Acceder a los elementos de una matriz

MAT

7	8
5	4
6	9
3	5



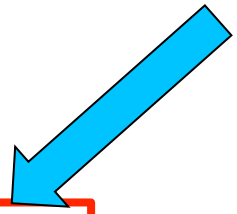

```
        ORG 2000
MAT     DATA 7
        DATA 8
        DATA 5
        DATA 4
        DATA 6
        DATA 9
        DATA 3
        DATA 5
```

Ejemplo 7: Acceder a los elementos de una matriz

- Pseudocódigo (versión con punteros):

```
CONT = 5;
Mientras (CONT!=0)
    FIL = LeerTeclado();
    COL = LeerTeclado();
    P = MAT + FIL*NCOL + COL;
    VAL = *P;
    EscribirPantalla(VAL);
    CONT = CONT-1;
Fin Mientras
```

Almacenamiento por filas



Ejemplo 7: Acceder a los elementos de una matriz

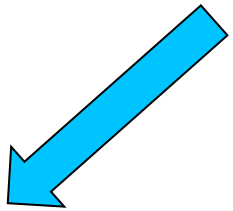

```
        ORG 0
        BR /INI
NFIL    DATA 3
NCOL    DATA 2
DMAT    DATA MAT
DCP     DATA 4092
DDP     DATA 4093
DCT     DATA 4094
DDT     DATA 4095
CONT    RES 1
P       RES 1
FIL     RES 1
COL     RES 1
VAL     RES 1
INI     LD.A, #1
        ST.A, [/DCP]
        LD.A, #5
        ST.A, /CONT
BUCLE   BZ /FBUCLE
LEE_F   LD.A, [/DCT]
        BZ /LEE_F
        LD.A, [/DDT]
        SUB.A, #48
        ST.A, /FIL
LEE_C   LD.A, [/DCT]
        BZ /LEE_C
        LD.A, [/DDT]
        SUB.A, #48
        ST.A, /COL
        LD .A, #0
        LD .X, /FIL
MUL     BZ /FMUL
        ADD .A, /NCOL
        SUB .X, #1
        BR /MUL
FMUL    ADD.A, /DMAT
        ADD.A, /COL
        ST.A, /P
        LD.A, [/P]
        ST.A, /VAL
ESC_P   LD.A, [/DCP]
        BZ /ESC_P
        LD.A, #48
        ADD.A, /VAL
        ST.A, [/DDP]
        LD.A, /CONT
        SUB.A, #1
        ST.A, /CONT
        BR /BUCLE
FBUCLE  HALT
        END
```

Ejemplo 7: Acceder a los elementos de una matriz

- Pseudocódigo (versión con puntero e indexado):

```
CONT = 5;
Mientras (CONT!=0)
        FIL = LeerTeclado();
        COL = LeerTeclado();
        P = MAT + FIL*NCOL;
        VAL = P[COL];
        EscribirPantalla(VAL);
        CONT = CONT-1;
Fin Mientras
```

Almacenamiento por filas



Ejemplo 7: Acceder a los elementos de una matriz


```
        ORG 0
        BR /INI
NFIL    DATA 3
NCOL    DATA 2
DMAT    DATA MAT
DCP     DATA 4092
DDP     DATA 4093
DCT     DATA 4094
DDT     DATA 4095
CONT    RES 1
P       RES 1
FIL     RES 1
COL     RES 1
VAL     RES 1
INI     LD.A, #5
        ST.A, /CONT
BUCLE   BZ /FBUCLE
LEE_F   LD.A, [/DCT]
        BZ /LEE_F
        LD.A, [/DDT]
        SUB.A, #48
        ST.A, /FIL
LEE_C   LD.A, [/DCT]
        BZ /LEE_C
        LD.A, [/DDT]
        SUB.A, #48
        ST.A, /COL
        LD .A, #0
        LD .X, /FIL
MUL     BZ /FMUL
        ADD .A, /NCOL
        SUB .X, #1
        BR /MUL
FMUL    ADD.A, /DMAT
        ST.A, /P
        LD.X, /COL
        LD.A, [/P][.X]
        ST.A, /VAL
ESC_P   LD.A, [/DCP]
        BZ /ESC_P
        LD.A, #48
        ADD.A, /VAL
        ST.A, [/DDP]
        LD.A, /CONT
        SUB.A, #1
        ST.A, /CONT
        BR /BUCLE
FBUCLE  HALT
        END
```

Ejemplo 8: Modificación de Símplez+I^3

C16. Se tiene una máquina Símplez+i^3 con las siguientes modificaciones:

- El bus de direcciones es de 14 bits.
- El único modo de direccionamiento posible es el directo.

Se pide responder **RAZONADAMENTE** a las siguientes cuestiones:

a) ¿Cuál es la máxima capacidad de la memoria principal de esta máquina?

b) Indicar cuál tiene que ser el formato de las instrucciones para que se pueda aprovechar dicha capacidad máxima.

c) ¿Cuál sería el ancho de las palabras de memoria?

d) ¿Cuántos bits debería tener el acumulador? ¿y el registro X?