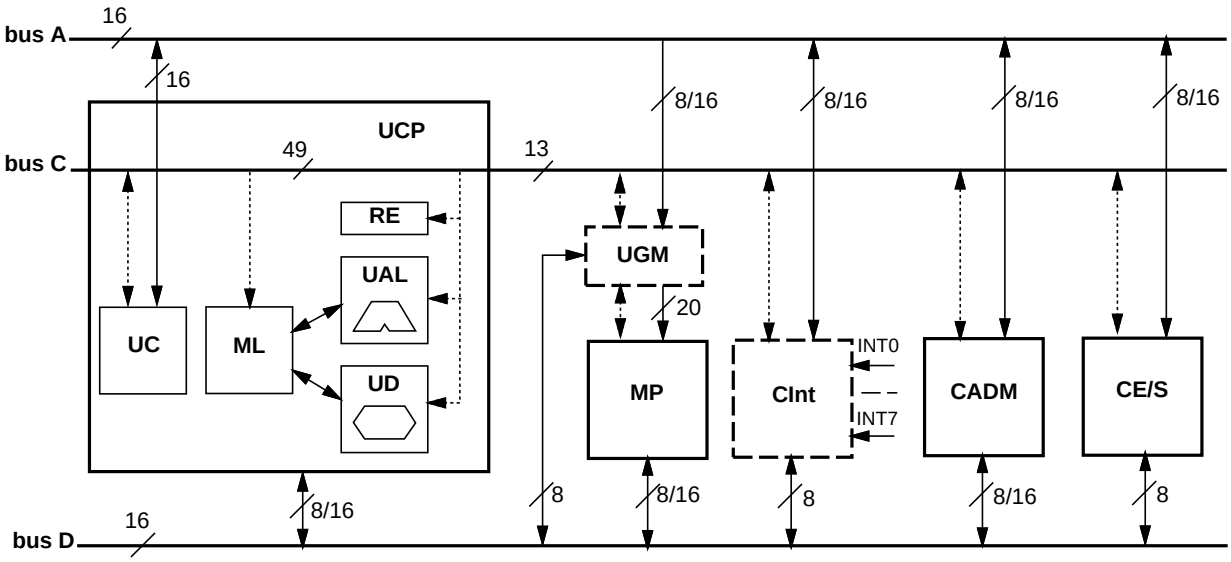
Resumen de Algorítmez

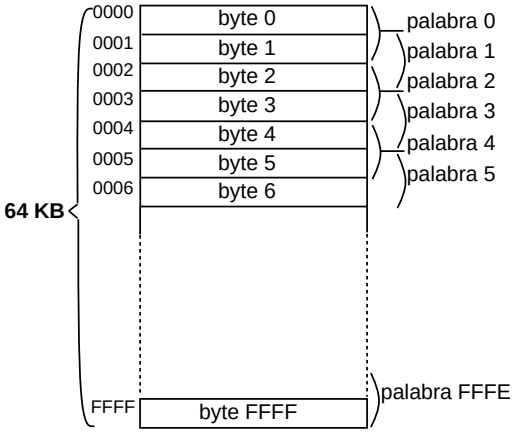
1. Modelo estructural
2. Instrucciones y modos de direccionamiento
3. Lenguaje ensamblador
4. Interrupciones
5. Software para el servicio de interrupciones
6. Controlador de ADM
7. Unidad de gestión de memoria

Modelo estructural

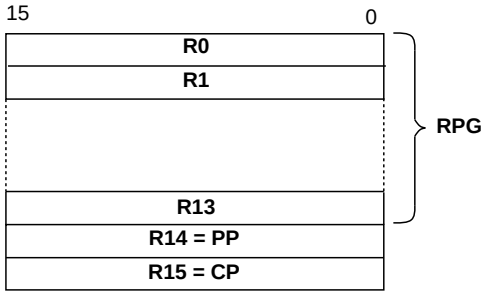


Memoria y registros

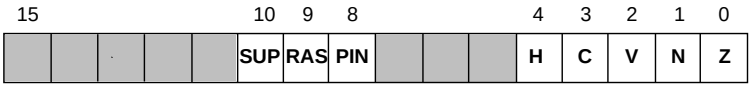
♣ MP (vista por la UCP):



♣ ML:



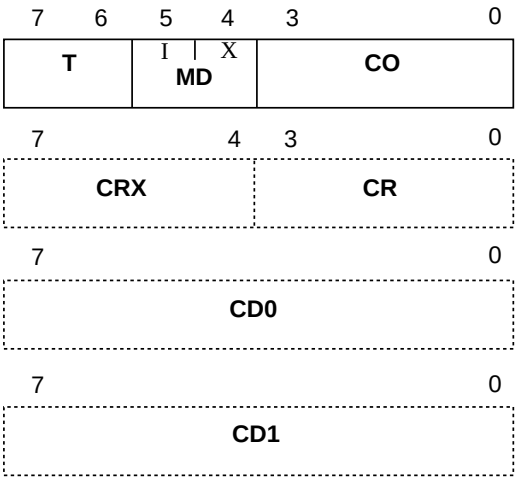
♣ Registro de estado:



Formatos y convenios de representación

- **Números enteros:**
  - 8 ó 16 bits
  - Complemento a 2
  - Extremista menor:
    - LSB en d
    - MSB en d+1
- **Cadenas de caracteres:**
  - ISO Latin1 (o ISO Latin9)

- **Instrucciones:**



Instrucciones importantes

- `CALL`: (PP)–2 → PP; (CP) → (PP); DE → CP
- `RET`: ((PP)) → CP; (PP)+2 → PP
- `BRK`: Genera interrupción de programa;
  como consecuencia:
  (PP)–2 → PP; (CP) → (PP);
  (PP)–2 → PP; (RE) → (PP);
  0 → PIN; 1 → SUP;
  (260) → CP (vector de interrupción de `BRK`)
- `RETI`: 0 → SUP
  ((PP)) → RE; (PP)+2 → PP;
  ((PP)) → CP; (PP)+2 → PP

Instrucciones privilegiadas

- `EI, DI, LD .E, RETI`
- `IN, OUT`
- `WAIT, HALT`

Su intento de ejecución cuando (SUP)=0 provoca una *interrupción por violación del modo usuario*

Modos de direccionamiento

MD	Nombre	Efecto	Resultado para (CRX)=H'F
00	autoincremento	DE=(RX) (RX)+2→ RX ó (RX)+1→ RX	inmediato (con CD de 1 o 2 bytes)
01	indexado	DE=(CD)+(RX) CD: 1 byte con signo	relativo a CP
10	autoincremento indirecto	DE=((RX)) (RX)+2→RX	directo (con CD de 2 bytes)
11	indexado indirecto	DE=((CD)+(RX)) CD: 1 byte con signo	relativo a CP e indirecto

Lenguaje ensamblador (IEEE 694)

- Seudoinstrucciones: `RES`, `RES.B`, `DATA`, `DATA.B`
- Directivas:
  - `ORG <dir>`
  - `EQU <cte, símb. o expr.>`
  - `MODULE <nombre>`
  - `FROM <nombre> IMPORT <simb1>,<simb2>,...`
  - `EXPORT <simb1>, <simb2>, ...`
  - `END`, o `END <etiqueta>`
- Expresiones:
  - Se construyen con símbolos, «+», «–», «(» y «)»
  - Se evalúan en tiempo de traducción

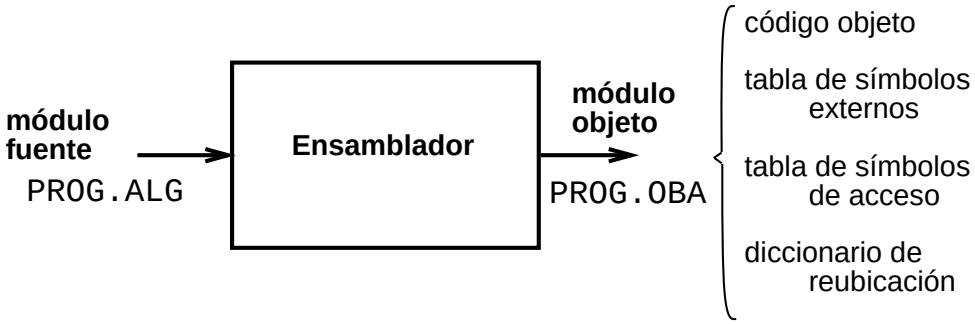
Lenguaje ensamblador: direccionamiento (1)

Modo	Ejemplos	Efectos
auto-incremento	`LD.B .0,[.3++]`	((R3))$\rightarrow_B$R0; (R3)+1$\rightarrow$R3
	`BR [.3++]`	(R3)$\rightarrow$CP; (R3)+2$\rightarrow$R3
indexado	`LD.B .0,/ETI[.3]`	(d(`ETI`)+(R3))$\rightarrow_B$R0
	`BR /ETI[.3]`	d(`ETI`)+(R3)$\rightarrow$CP
auto-incremento indirecto	`LD.B .0,[[.3++]]`	(((R3)))$\rightarrow_B$R0; (R3)+2$\rightarrow$R3
	`BR [[.3++]]`	((R3))$\rightarrow$CP; (R3)+2$\rightarrow$R3
indexado indirecto	`LD.B .0,[/ETI[.3]]`	((d(`ETI`)+(R3)))$\rightarrow_B$R0
	`BR [/ETI[.3]]`	(d(`ETI`)+(R3))$\rightarrow$CP

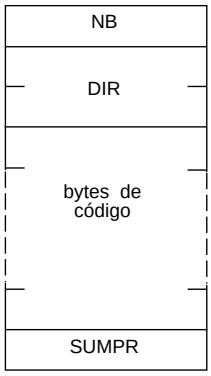
Lenguaje ensamblador: direccionamiento (2)

Modo	Ejemplos	Efectos
inmediato	`LD.B .0,#-125`	H'83$\rightarrow_B$R0 (8 bits)
	`BR #-125`	(absurdo)
relativo a programa	`LD.B .0,$ETI` o bien: `LD.B .0,ETI`	((CP)+drel(`ETI`))$\rightarrow_B$R0
	`BR $ETI` o bien: `BR ETI`	(CP)+drel(`ETI`)$\rightarrow$CP
directo	`LD.B .0,/ETI`	(d(`ETI`))$\rightarrow_B$R0
	`BR /ETI`	d(`ETI`)$\rightarrow$CP
relativo e indirecto	`LD.B .0,[$ETI]` o bien: `LD.B .0,[ETI]`	(((CP)+drel(`ETI`)))$\rightarrow_B$R0
	`BR [$ETI]` o bien: `BR [ETI]`	((CP)+drel(`ETI`))$\rightarrow$CP

Programa ensamblador



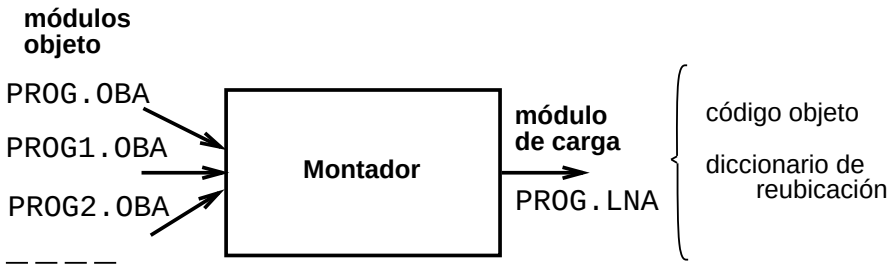
Registro de código:

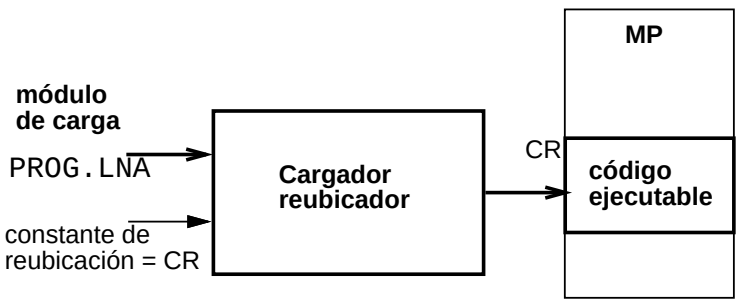


Formato del fichero PROG.OBA:

- Cabecera *(header)*: número mágico + nombre
- El resto, como una sucesión de registros *(records)* de longitud variable

Montador y cargador





Entrada/Salida

Iniciación del teclado (puertos 0 y 1)
y la pantalla (puertos 2 y 3):

```
----
LD.B    .0, #0
LD.B    .1, #1
LD.B    .2, #2
LD.B    .3, #3
CLR     .4
OUT     .4, [.0];  0->PRE
LD.B    .4, #1
OUT     .4, [.2];  1->PRS
----
```


Espera activa

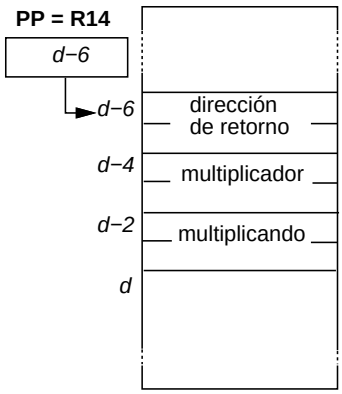
Rutinas para lectura y escritura de caracteres:

```
LEECAR  IN    .4, [.0] ;comprueba
        AND   .4, #1   ; estado
        BZ     LEECAR  ; teclado y
        IN    .5, [.1] ; lee
        CALL   ESCCAR  ; hace eco
        RET
ESCCAR  IN    .4, [.2] ;comprueba
        AND   .4, #1   ; estado
        BZ     ESCCAR  ; pantalla y
        OUT   .5, [.3] ; escribe
        RET
        END
```

Subprogramas: paso de argumentos (o parámetros)

- Paso de valor o paso de referencia (dirección)
- Reserva estática (en MP) o dinámica (en registros o en pila)
- Ejemplo de paso de parámetros por la pila:

```
...
PUSH  .1; pasa el multiplicando
PUSH  .2; pasa el multiplicador
CALL  /MULT
POP   .3; recoge el resultado
POP   .4;  (32 bits)
...
```



¿Cómo los coge el subprograma?
¿Cómo queda la pila al salir de `MULT`*?*

Traducción de las llamadas a procedimientos

Programa en C:

```
int factorial(int num) {
  int fact = 1;
  do {
    fact = num*fact;
    num = num - 1;
  } while (num > 0);
  return fact;
}
  ...
int f;
/* llamada para 5!: */
f = factorial(5);
```

Resultado de la compilación:

```
FACT  LD    .1,/2[.14]
      LD    .2,#1
BUC   CALL  /MUL
      SUB   .1,#1
      BNZ   BUC
      RET
      ...
F     RES   1
      ...
      LD    .0,#5
      PUSH  .0
      CALL  /FACT
      ADD   .14,#2
      ST    .2,F
```

Causas de interrupción

- Externas (128)
  - Una sola línea de interrupción (enmascarable con `DI`)
  - Pueden permitirse o inhibirse individualmente poniendo IT (en el puerto de estado) a 1 o a 0
- No enmascarable, **NMI**
  - Otra línea de entrada a la UCP
- Internas
  - **VMU** (violación del modo usuario)
  - **BK** (instrucción `BRK`)
  - **BKV** (instrucción `BRKV`)
  - **RS** (si (RAS)=1)

Vectores de interrupción

	Causa	Dir (Dec./Hex.)
0	periférico 0	0/0
1	periférico 1	2/2
...	...	...
...	...	...
...	...	...
127	periférico 127	254/FE
128	rastreo	256/100
129	instrucción `BRKV`	258/102
130	instrucción `BRK`	260/104
131	no enmascarable	262/106
132	violación modo	264/108

Interrupciones: modelo procesal de la UCP

Cuando no enmascarable (NMI, VMU, BK o BKV) o bien otra y (PIN)=1:

1. Si VMU, aborta ejecución y pasa a 3
2. Termina de ejecutar la instrucción en curso
3. (PP)$-2 \rightarrow$PP; (CP)$\rightarrow$(PP);
   (PP)$-2 \rightarrow$PP; (RE)$\rightarrow$(PP);
4. 0$\rightarrow$PIN; 1$\rightarrow$SUP;
5. Si NMI o causa interna, pone contenido del vector en CP. Si no, genera señal, recoge dirección del vector por el bus A, lee contenido y lo pone en CP
6. Sigue ejecutando instrucciones

Esquemas para las interrupciones

Para explorar las causas externas, varias posibilidades hardware (externo a la UCP)/software:

♣ *Consulta por software:*

- V=D'266 a bus A *siempre* (hardware mínimo)
- La palabra 266 contiene la dirección de una **rutina de consulta de interrupciones**

♣ *Consulta por hardware:*

La causa externa pone la dirección de su vector en el bus A

♣ *Controlador de interrupciones:*

Gestiona varias líneas de interrupcion y genera la dirección del vector

Consulta mediante software

- El hardware externo (**mínimo**) pone D'266 en el bus A; si (266)=X, en [MP(X)] empieza la RCI
- La RCI consulta puertos de estado hasta encontrar (PR)=(IT)=1
- **Variantes:**

  (a) De RSi a A

  (b) De RSi a B (prioridad circular, o «round robin»)

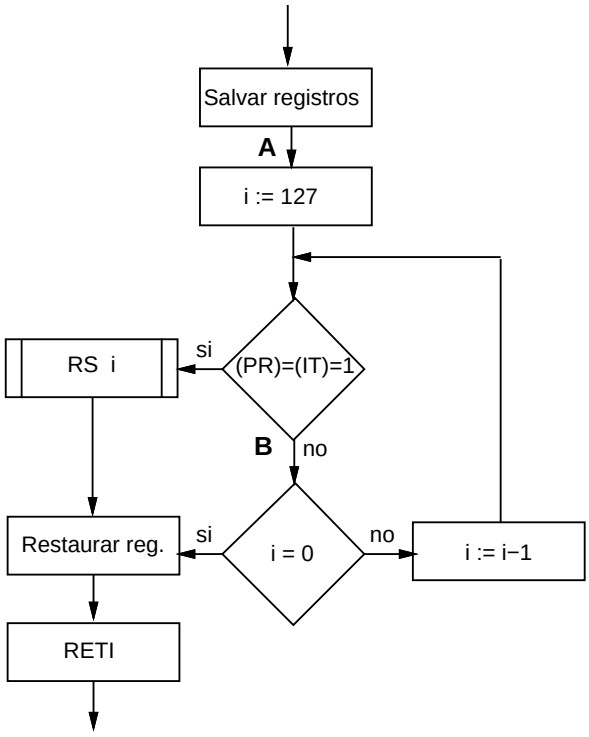


Tabla de periféricos:

La construye el S.O. al arrancar, y contiene informaciones para el S.O.

Ejemplo:

dir MP	+0	+1	+2	+3	...	+19
d	12	Aϕ	ϕ	ϕ	...	ϕ
d+20	10	5ϕ	ϕ	ϕ	...	ϕ
d+40	0E	0ϕ	ϕ	ϕ	...	ϕ
d+60	02	8ϕ	ϕ	ϕ	...	ϕ
d+80	00	8ϕ	ϕ	ϕ	...	ϕ
d+100	FF	ϕ	ϕ	ϕ	...	ϕ

H'12 H'02 y H'00 operativos; H'10 y H'0E no operativos

La RCI explora esta tabla de arriba abajo

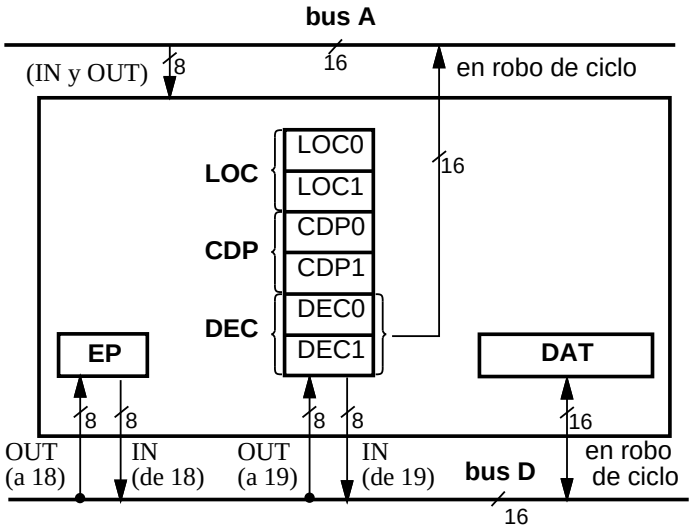
Rutina de consulta de interrupciones (1)

```
   MODULE  RCI
   EXPORT  RCI_ENT          ; dir. relativa de comienzo
   FROM  GP IMPORT  TBPRF ; dir. tabla periféricos
MSKOP     EQU       H'0080    ; para ver si operativo
MSKIT     EQU       H'0003    ; para ver si (PR)=(IT)=1
MSKFIN    EQU       H'FF      ; para ver si fin tabla
RCI_ENT   PUSH      .0        ; guarda los registros
          PUSH      .1        ; que va a utilizar:
          PUSH      .2        ; R0, R1 y R2
          LD        .0,#TBPRF; R0: puntero a la tabla
          CLR       .1        ; R1: puntero al vector
BUCLE     LD.B      .1, [.0++] ; carga byte0 en R1
          CMP.B     .1, #MSKFIN; mira si fin de tabla
          BZ        FIN
```

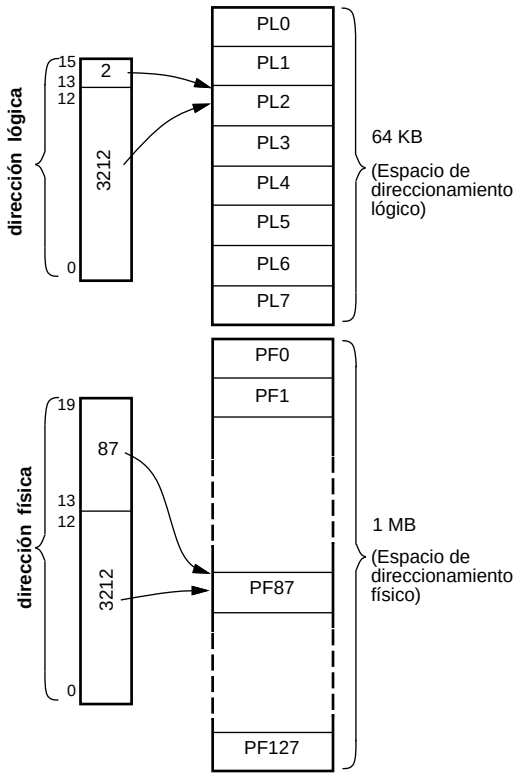

Rutina de consulta de interrupciones (2)

```
          LD.B      .2, [.0++] ; carga byte1 en R2
          AND       .2, #MSKOP ; mira si operativo
          BZ       SIGUE
          IN        .2, [.1]   ; lee estado periférico
          NOT         .2
          AND       .2, #MSKIT ; mira si interrupción
          BNZ        SIGUE     ;    pendiente
          CALL      [[.1++]]   ; bifurca a RS
FIN       POP       .2         ; restaura los registros
          POP       .1
          POP       .0
          RETI                 ; retorno de interrupción
SIGUE     ADD       .0, #18 ; actualiza R0 para que
          BR        BUCLE   ;  apunte al byte0 siguiente
         END
```

Controlador de acceso directo a la memoria



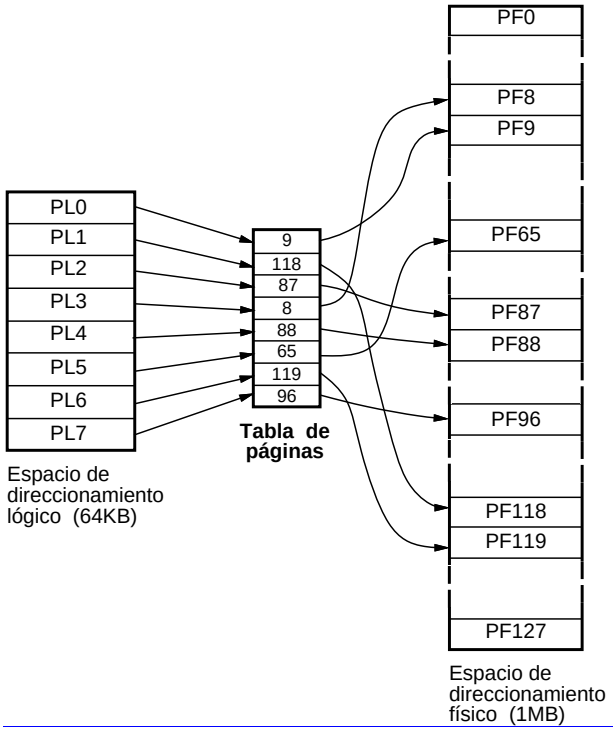
Gestión de memoria en Algorítmez



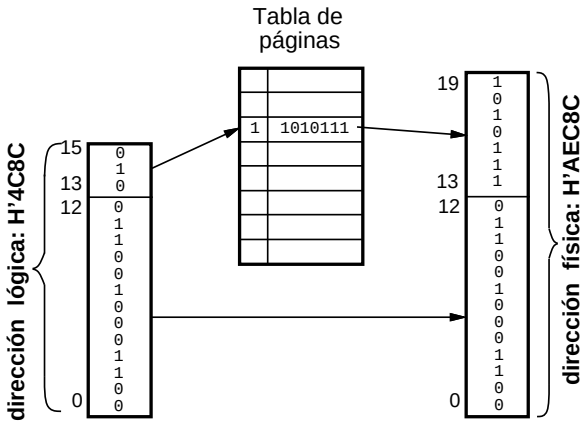
- **Normalmente**,
  espacio lógico *(virtual)*
  ≫ espacio físico *(real)*
  ⇒ *memoria virtual*
- En Algorítmez,
  espacio lógico = 64 KB;
  espacio físico = 1 MB

Traducción mediante tabla de páginas

Función de correspondencia:



Ejemplo de traducción:



Unidad de gestión de memoria

- *Traducción:*
  dir. lógica ⇒ Ti ⇒ dir. física
  (si bit validez = 0, interrupción)
- *Cambio de func. corresp.:*
  8 `OUT` sobre puerto 21
  0 → bit C provoca {P} → {T}
- *func. corresp. para el S.O:*
  1 → bit C provoca:
  H'80 → T0, H'81 → T1. . .
- *cambio automático al cambiar* ***modo****:*
  (modo=0 ⇒ modo=1) ≡ (1 → C)
  (modo=1 ⇒ modo=0) ≡ (0 → C)