UTN.BA
UNIVERSIDAD TECNOLÓGICA NACIONAL
FACULTAD REGIONAL BUENOS AIRES

Microprocesadores
Interrupciones

Profesora Silvana Panizzo

Repaso
Ciclo de instrucción

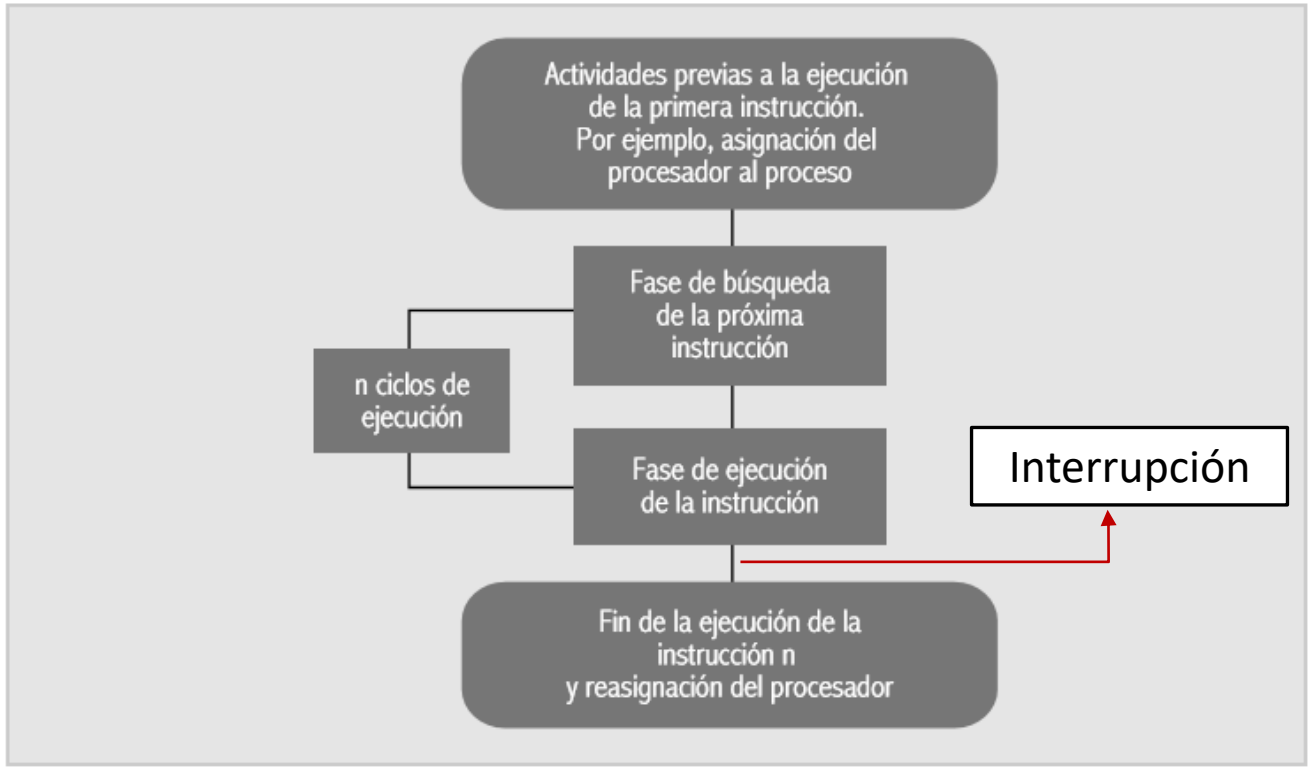


Fig. 8.2. Ciclo de instrucción para las n instrucciones de un programa.

Fase de búsqueda:

- Calculo de la dirección física de la instrucción.
- Dar orden de lectura RD
- Se carga el registro IR

Fase de ejecución:

- Interpretar el código de la instrucción (Decodificar)
- Incrementar el IP
- Búsqueda del dato **(RD)** o Guardar el dato **(WR)** (si afecta)
- Generar orden al modulo para que opere el dato.

Capacidad de interrupción

- ✓Las interrupciones y excepciones son acontecimientos que provocan un desvío en el flujo de control de la CPU.
  - ✓Interrupciones son acontecimientos externos que desvían el flujo de control (por ejemplo, averías en el hardware)
  - ✓Excepciones son internas y se producen como consecuencia de una anomalía dentro de la CPU (por ejemplo, una división por cero) durante la ejecución de un programa.

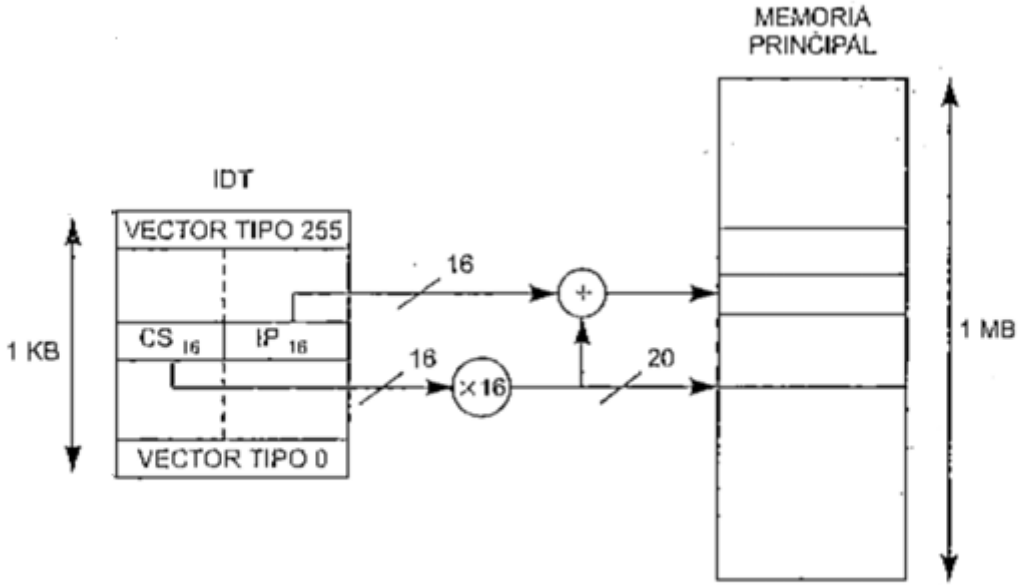
UTN.BA
UNIVERSIDAD TECNOLÓGICA NACIONAL
FACULTAD REGIONAL BUENOS AIRES

IDT – Modo real

- ✓Para el manejo de interrupciones y excepciones, el modelo dispone de una tabla o vector de interrupciones llamada tabla de descriptores de interrupciones y excepciones (IDT).
- ✓La IDT esta formada por 256 entradas, cada uno de los diferentes tipos de interrupciones esta asociado a una de esas entradas.
- ✓La IDT ocupa un segmento, cuya base y limite están contenidos en el Registro de Tablas de descriptores de Interrupciones (IDTR).
- ✓Tanto en modo protegido como en modo real se utiliza la IDT, aunque la forma de operar es distinta en cada modo, el objetivo es el mismo.

IDT – Modo real

- ✓ En la inicialización o reset del modelo (modo real), el SO carga la base del registro IDTR con el valor 0000 0000_H, ya que solo se utiliza el primer MB de la memoria.
- ✓ Se trabaja con segmentos (no existe la paginación) y máximo tamaño del registro es de 64 KB.
- ✓ En cada entrada de la IDT, se apunta al segmento de código donde reside la rutina que atiende a la interrupción.
- ✓ El tamaño máximo de la IDT es de 1 KB, o sea, 256 entradas x 4 bytes/entrada, por lo tanto el limite es $03FF_H$



IDT - Vector

Relación entre el numero de vector y su función.

Hay tantas entradas en la IDT como servicios definidos, cada vector (puntero) contiene la posición del servicio.

Función	Número de vector
Error de división	0
Interrupción paso a paso	1
Interrupción NMI	2
Interrupción punto de parada	3
Excepción por sobrepasamiento	4
Sobrepasamiento de los límites	5
Código OP inválido	6
Coprocesador no disponible	7
Reservadas (Modo protegido)	8-15
Error de coprocesador	16
Reservados INTEL	17-31
Definidas por el usuario	32-255

Figura 14.9. *Relación de vectores de interrupción comunes en modo Real y modo Protegido.*

Concepto de pila

La pila es una estructura de datos en memoria de acceso **LIFO (Last in first out).**

El registro de segmento SS, se accede con criterio LIFO. Imaginar como una pila de platos.

La CPU utiliza la pila para:

- Almacenar la dirección de retorno (IP), y eventualmente el CS cuando se llama aun procedimiento o subrrutina.
- Almacena el estado del procesador cuando se produce una interrupción. Los registros obligados que apila con el CS, el IP y el estado de los flags y eventualmente los registros de calculo, etc.
- Para pasar parámetros entre dos procedimientos.

El acceso a la pila se realiza a través de los registros punteros SP y BP, donde SP contiene la dirección del próximo elemento de la pila vacio. Es decir SS:SP

Concepto de pila

La escritura o lectura de datos de la pila es un procedimiento software y se realizan incrementado o decrementado el registro SP.

Si la pila trabaja a nivel palabra (2 bytes) , cada vez que se extraiga un dato se autoincrementara el SP = SP + 2 y cada vez que se cargue se autodecrementara (SP = SP – 2) en 2 unidades.

La encargada de acceder a la pila es la CPU, ejecutando instrucciones propias para esta estructura, como **PUSH y POP**.

- PUSH, almacena una palabra en la pila y luego decrementa SP
- POP, extrae una palabra de la pila y desafecta las posiciones de memoria incrementando el SP.

Las instrucciones que a causa de su ejecución afectan a la pila son **CALL, INT, RET o IRET**.

- **CALL y Ret** son instrucciones que sirven para invocar y dar el retorno a un procedimiento o subrutina.
- **INT e IRET** cumplen la misma función cuando se invoca una subrrutina de interrupción.

Tipos de interrupciones y excepciones

Tabla 8-6. Diferencias entre distintos tipos de interrupciones.

Externas o hardware: son convocadas en forma asincrónica, no dependen del programa en ejecución.	No enmascarables (NMI) La CPU es avisada por una señal de control llamada comúnmente NMI (*Non Mascarable Interrupt*).	Siempre son atendidas. Se consideran de máxima importancia, pues las provoca el hardware ante acontecimientos que no pueden ser por lo menos "avisados" (p. ej., la caída de tensión de la alimentación).
	Enmascarables La CPU es avisada por otra señal que la diferencia de la NMI.	Se consulta la señal de interrupción por cada ciclo de ejecución de instrucción. Como no siempre debe ser atendida, se consulta también una bandera de estado de habilitación de interrupciones. Si la bandera lo autoriza, entonces la CPU suspende de manera momentánea la ejecución de programa activo y ejecuta el servicio de interrupción. Una función hardware externa, determina la **prioridad** que tiene esta interrupción respecto de otras peticiones pendientes y le da curso (p. ej., recepción de un byte en un puerto). Si la bandera de interrupción está desactivada, la CPU no la tiene en cuenta y continúa la ejecución del programa activo; de ahí deriva el nombre de "mascarable" o "enmascarable".

Tipos de interrupciones y excepciones

Internas o software: son convocadas por el programa.		Se pueden producir por la ejecución de una instrucción específica dentro de un programa, para solicitar una interrupción que cumpla con alguna función determinada. La forma de convocarla es INT #, donde # corresponde al número que identifica la interrupción.
Excepciones: son provocadas como consecuencia de anomalías que se producen y detectan durante la ejecución del programa y a causa de ella.	Faltas o errores	Son las que se pueden detectar y corregir antes de que se produzca la ejecución de una instrucción determinada (p. ej., se produce una falla al invocar una página que no está en memoria principal). La atención de esta interrupción provoca que el sistema operativo localice la página faltante y la cargue.
	Trampas	Son las que se detectan una vez ejecutada la instrucción que las provoca (p. ej., las que de alguna manera incorpora el usuario en el programa, como sobreflujo u *overflow*).
	Abortos	Son las que se detectan sin localizar la instrucción que las provoca, abortando la ejecución del programa (p. ej., un valor no válido en un registro de sistema).

Fases de atención de una interrupción o excepción

✓**Fase 1**

Se comprueba si hay interrupciones pendientes para ser atendidas. Si hay varias peticiones de interrupciones pendientes, se atenderán por orden de prioridad.

✓**Fase 2**

Al comenzar la interrupción se **resguarda el entorno actual de la CPU** (context switch) para que al finalizar la interrupción o excepción se retorne al mismo punto, es decir, se guardan el contenido de los registros del procesador en la pila.

Por ultimo IF = 0 prohíbe las interrupciones mascarables, evitando que una interrupción de menor importancia interrumpa a la rutina en ejecución. (ejemplo, periférico)

✓**Fase 3**

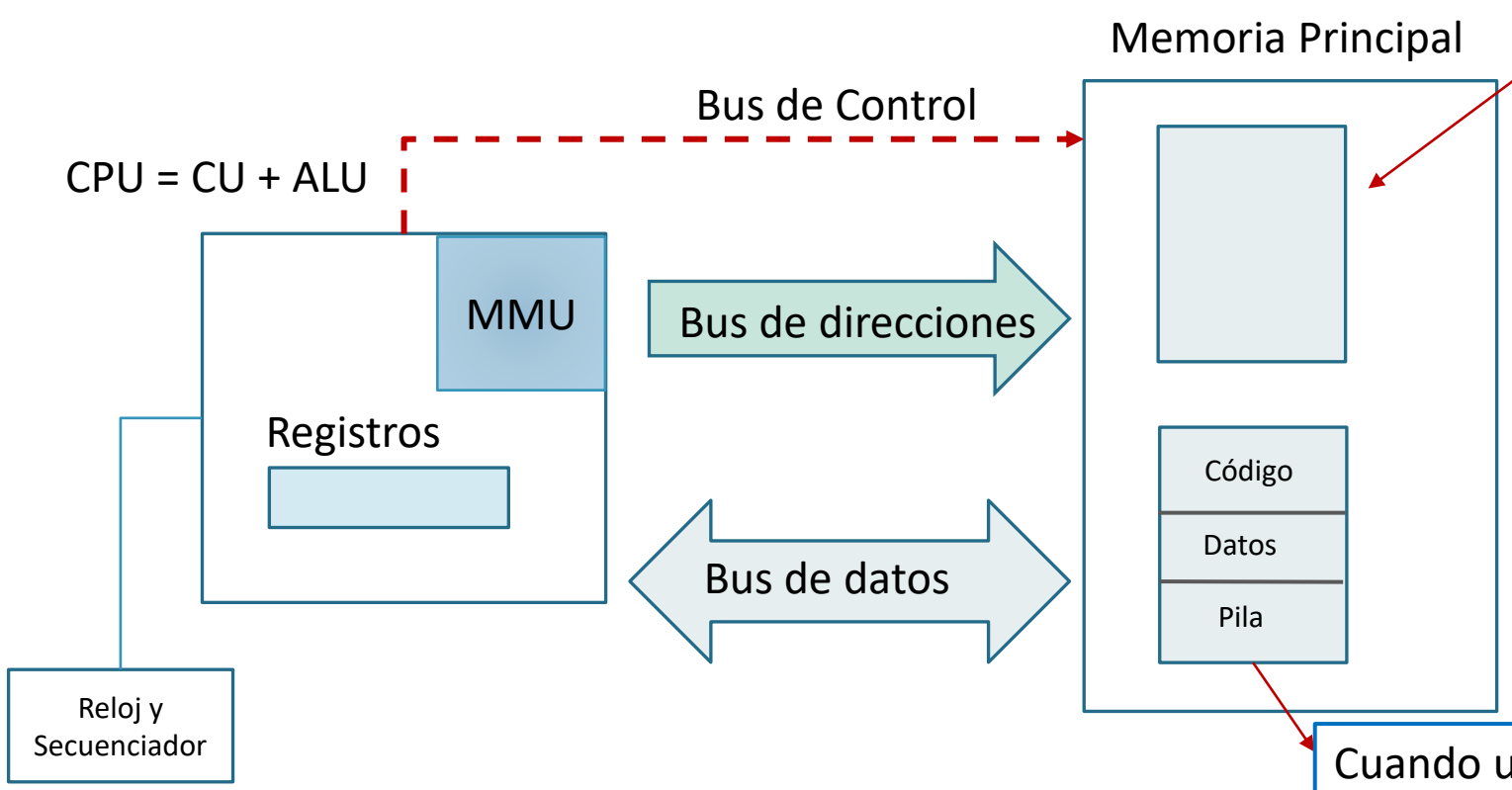
Se busca la entrada correspondiente en la IDT y se cargan los registros CS e IP con el valor contenido en dicha entrada, de esta forma, se obtiene el inicio de la rutina y comienza la ejecución de la misma.

✓**Fase 4**

La rutina de interrupción finaliza con la instrucción IRET, **se restaura el contexto de la CPU** (context switch). De esa manera, el programa continua su ejecución en la siguiente instrucción a la que se produjo la interrupción o excepción.

El flag IF=1, es decir, se vuelven a activar las interrupciones mascarables.

Repaso
CPU – Memoria Principal – Modo real

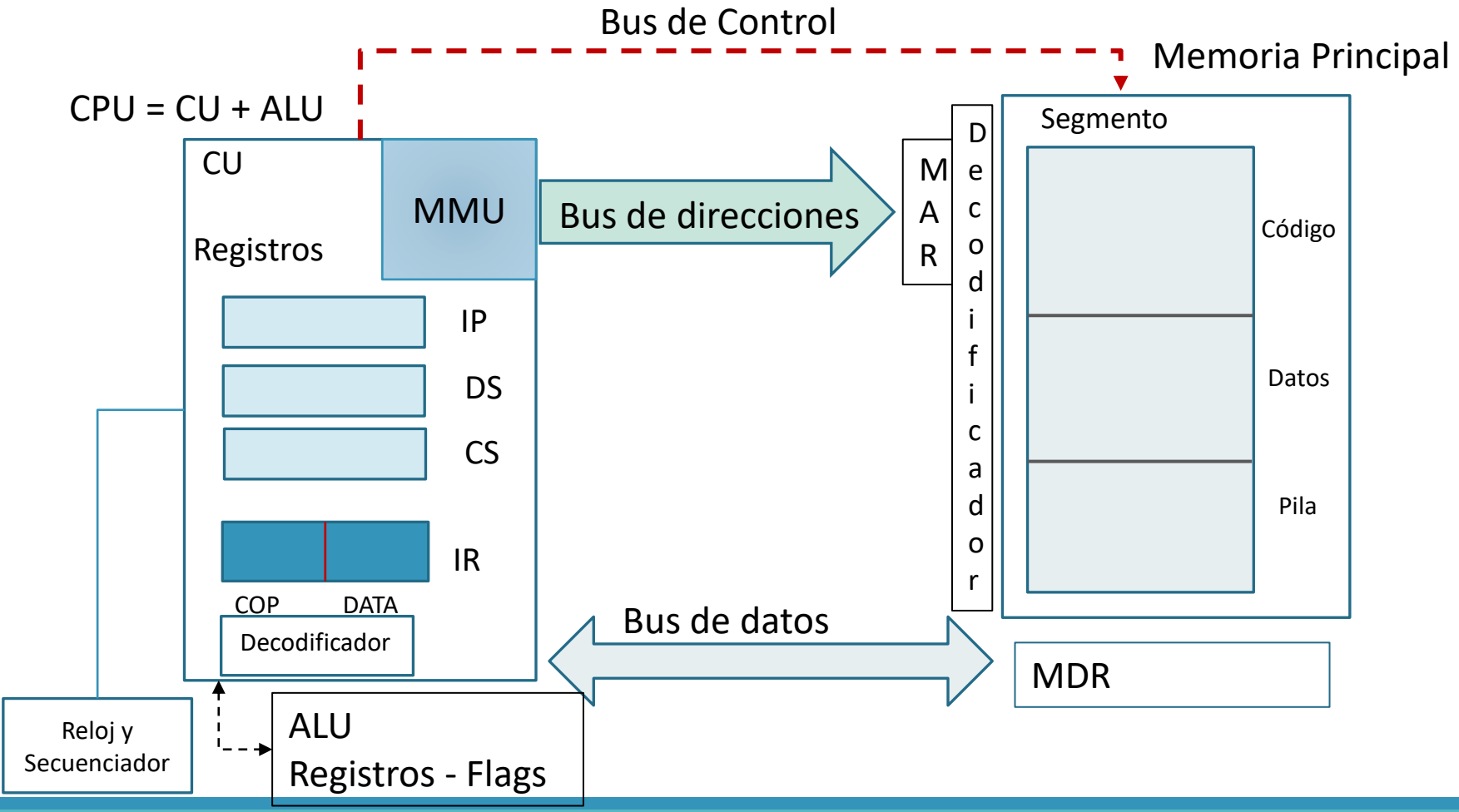


Segmento

- ✓ Para identificar un dato o instrucción dentro de la memoria se debe obtener la dirección física.
- ✓ La dirección física se obtiene a partir de la dirección lógica.
- ✓ La dirección lógica se compone de la **Base del segmento : Desplazamiento**

Cuando un segmento es asignado para crear un programa en Assembler, el segmento se divide lógicamente en 3 partes: Código, Datos y Pila.

Repaso
CPU – Memoria Principal – Modo real

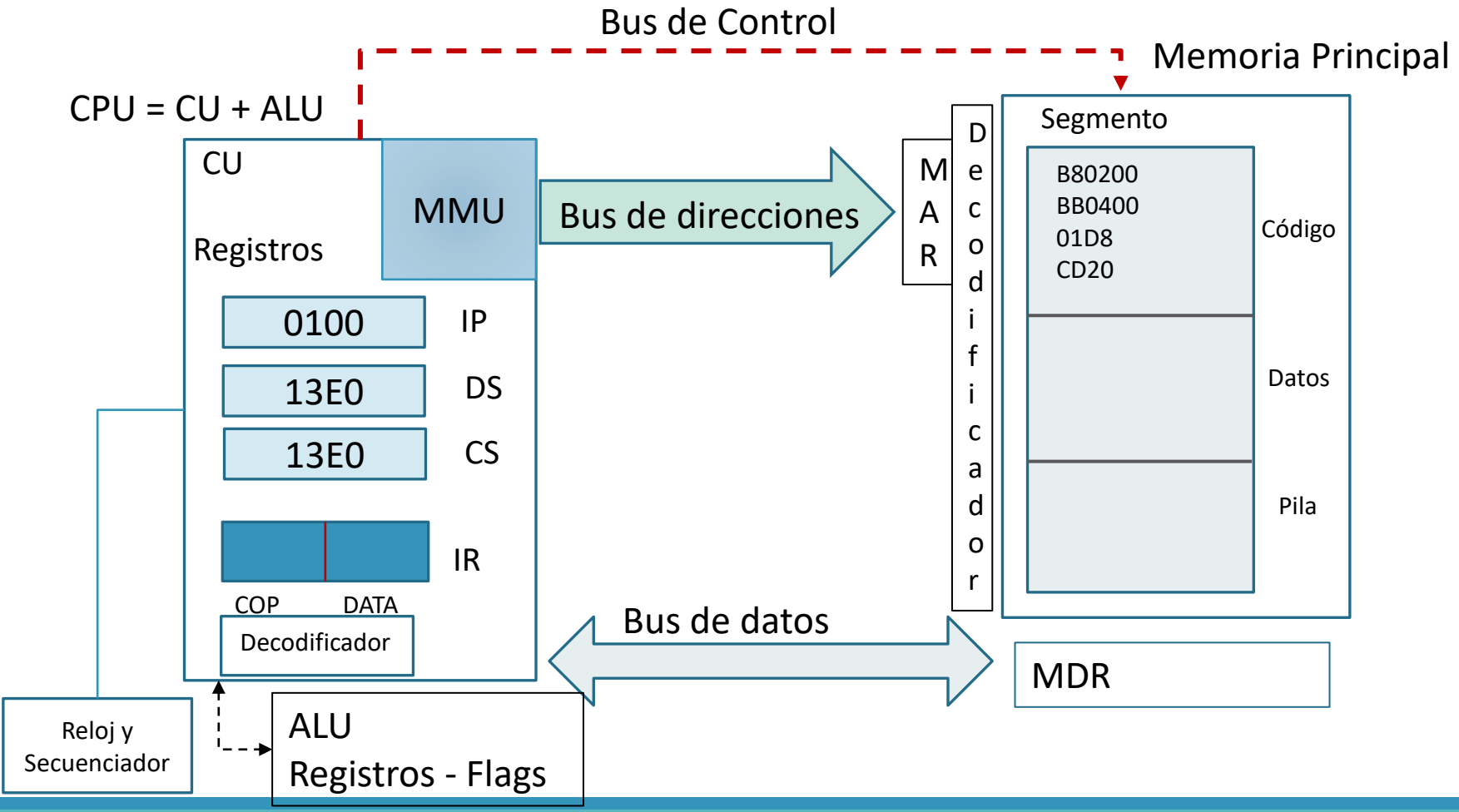


Ejemplo 1.txt

```
-a
13E0:0100 mov ax,0002
13E0:0103 mov bx,0004
13E0:0106 add ax,bx
13E0:0108 int 20
13E0:010A

-u
13E0:0100 B80200   MOV    AX,0002
13E0:0103 BB0400   MOV    BX,0004
13E0:0106 01D8     ADD    AX,BX
13E0:0108 CD20     INT    20
```

Repaso
Ciclo de instrucción – Actividades previas

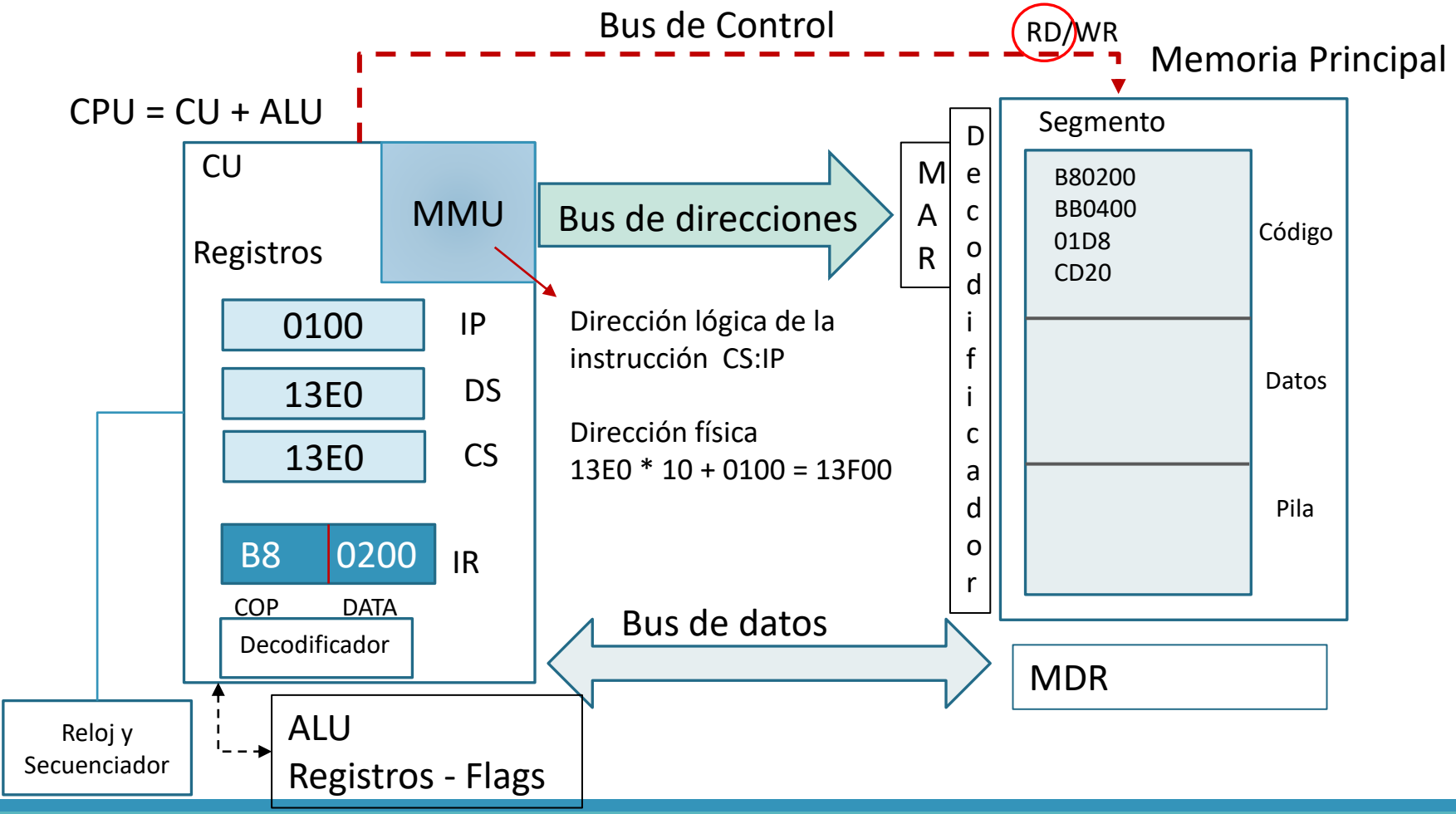


Actividades previas a la ejecución de la primer instrucción

```
-a
13E0:0100 mov ax,0002
13E0:0103 mov bx,0004
13E0:0106 add ax,bx
13E0:0108 int 20
13E0:010A

-u
13E0:0100 B80200   MOV    AX,0002
13E0:0103 BB0400   MOV    BX,0004
13E0:0106 01D8     ADD    AX,BX
13E0:0108 CD20     INT    20
```

Repaso
Ciclo de instrucción – Fase búsqueda

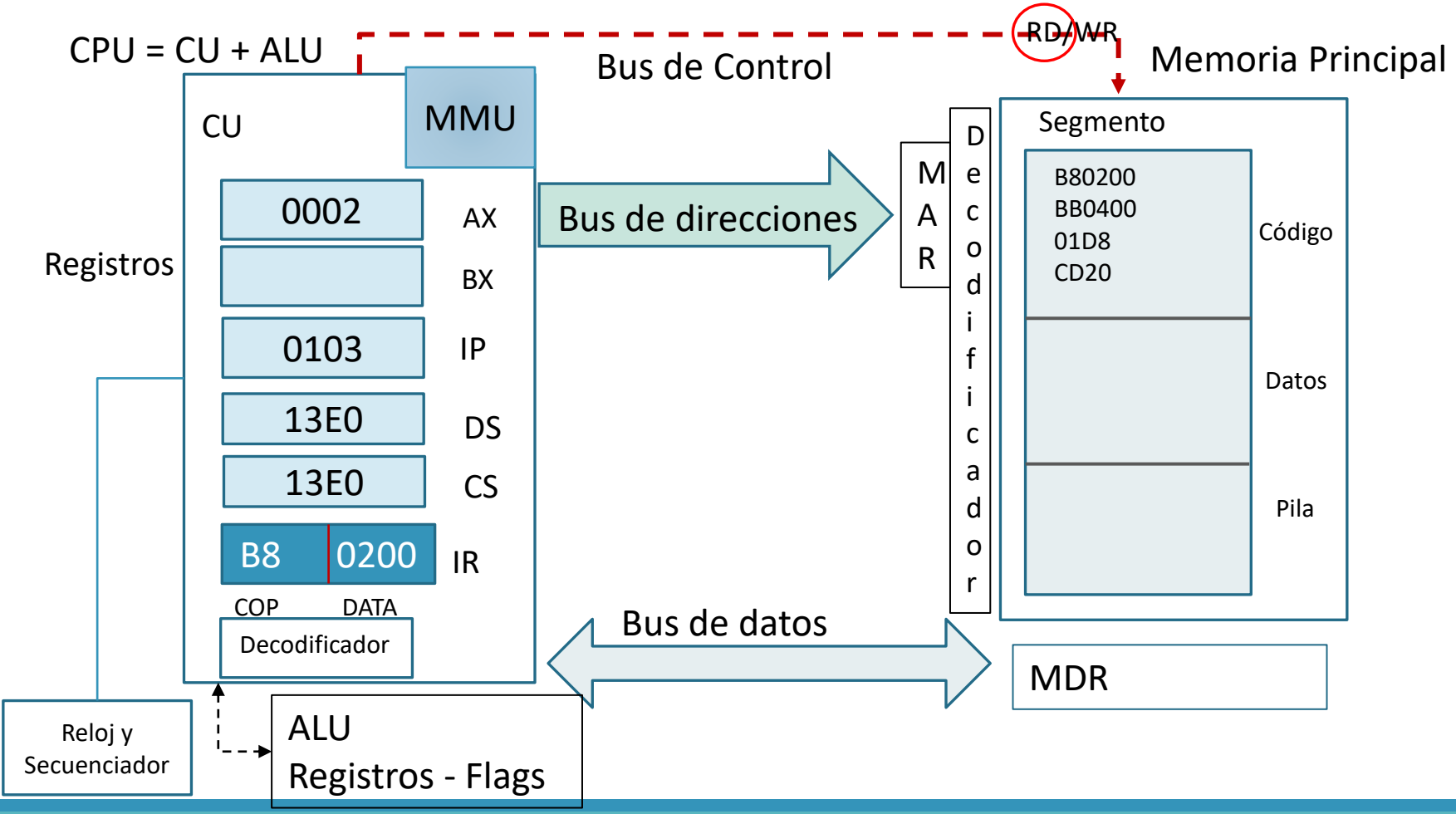


Fase de búsqueda:

- Calculo de la dirección física de la instrucción.
- Dar orden de lectura RD
- Se carga el registro IR

```
-a
13E0:0100 mov ax,0002
13E0:0103 mov bx,0004
13E0:0106 add ax,bx
13E0:0108 int 20
13E0:010A
-u
13E0:0100 B80200   MOV    AX,0002
13E0:0103 BB0400   MOV    BX,0004
13E0:0106 01D8     ADD    AX,BX
13E0:0108 CD20     INT    20
```

Repaso
Ciclo de instrucción – Fase ejecución



Fase de ejecución:

- Interpretar el código de la instrucción
- Incrementar IP
- Búsqueda del dato o Guarda el dato (si afecta)
- Generar orden al modulo para que opere el dato.

```
-a
13E0:0100 mov ax,0002
13E0:0103 mov bx,0004
13E0:0106 add ax,bx
13E0:0108 int 20
13E0:010A
-u
13E0:0100 B80200   MOV    AX,0002
13E0:0103 BB0400   MOV    BX,0004
13E0:0106 01D8     ADD    AX,BX
13E0:0108 CD20     INT    20
```

Repaso
Banderas y Registros

```
-r
AX=0000  BX=0000  CX=0000  DX=0000  SP=FFEE  BP=0000  SI=0000  DI=0000
DS=13E0  ES=13E0  SS=13E0  CS=13E0  IP=0100   NV UP EI PL NZ NA PO NC
13E0:0100 B80200       MOV     AX,0002
-t
AX=0002  BX=0000  CX=0000  DX=0000  SP=FFEE  BP=0000  SI=0000  DI=0000
DS=13E0  ES=13E0  SS=13E0  CS=13E0  IP=0103   NV UP EI PL NZ NA PO NC
13E0:0103 BB0400       MOV     BX,0004
```

Al finalizar cada trace, consulta por interrupciones externas. Si hay interrupciones se guarda el contexto del programa en la pila y se ejecuta el código correspondiente a la interrupción; si no hay interrupciones se sigue con la ejecución del programa.

```
-t
AX=0002  BX=0004  CX=0000  DX=0000  SP=FFEE  BP=0000  SI=0000  DI=0000
DS=13E0  ES=13E0  SS=13E0  CS=13E0  IP=0106   NV UP EI PL NZ NA PO NC
13E0:0106 01D8         ADD     AX,BX
```

Overflow
NV = no hay desbordamiento;
OV = sí lo hay

Direction
UP = hacia adelante;
DN = hacia atras;

Interrupts (IF)
DI = desactivadas;
EI = activadas

Sign
PL = positivo;
NG = negativo

Zero
NZ = no es cero;
ZR = sí lo es

Auxiliary Carry
NA = no hay acarreo auxiliar;
AC = hay acarreo auxiliar

Parity
PO = paridad non;
PE = paridad par;

Carry
NC = no hay acarreo;
CY = Sí lo hay