Clase 17 - Repaso I2

IIC2343 - Arquitectura de Computadores

Profesor:
- Felipe Valenzuela González
Correo:
frvalenzuela@alumni.uc.cl

Subrutinas

Subrutina: Requisitos

- Parámetro de Entrada
- Valor de retorno
- Llamada a la subrutina (salto de ida y de vuelta)

```
def encontrar_maximo(arreglo):
    largo_maximo = len(arreglo)

    maximo = arreglo[0]
    i = 0

    while i < largo_maximo:
        if arreglo[i] > maximo:
            maximo = arreglo[i]
        i += 1

    return maximo
```

Subrutina: Parámetro de Entrada y Valor Retorno

- Tenemos dos opciones:
- **Almacenamiento por Registro:** Muy limitado en la arquitectura actual
- **Almacenamiento por Memoria:** Ocupar espacio de la memoria para almacenar. El limitante es más lento pero dado el espacio es el que **utilizaremos**

Subrutina: Implementación

- Agregamos un **Mux DataIn** a la entrada de la Memoria
- Agregamos un **Mux PC** a la entrada del Program Counter
- ¿Cómo nos aseguramos que no choquen las variables?

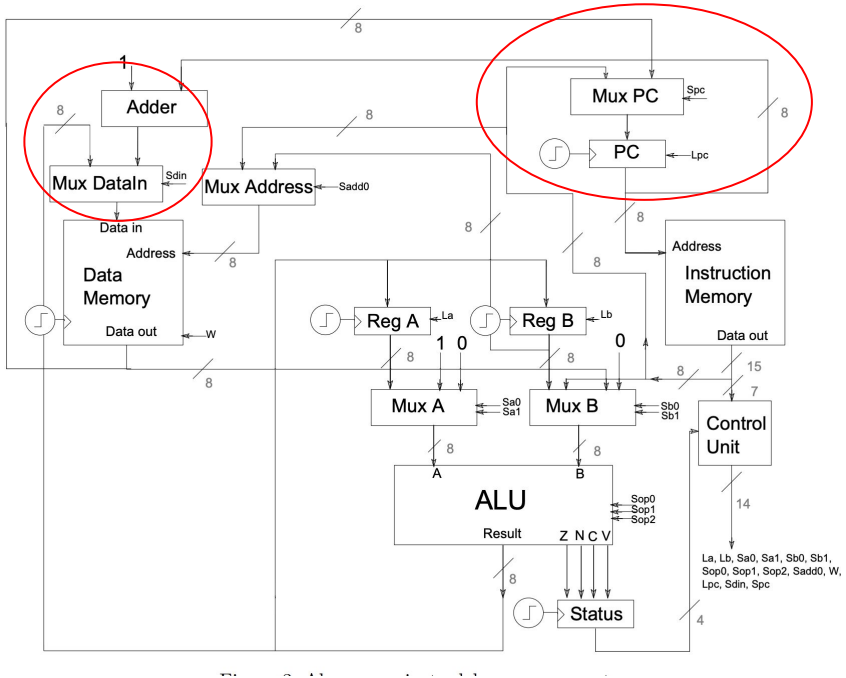

Imagen Referencia: https://github.com/IIC2343/Syllabus-anteriores/blob/main/Apuntes%20(viejos)/5%20-%20Saltos%20y%20Subrutinas.pdf

Subrutina: Assembly - Resumen

Instrucción	Operandos	Operación	Condiciones	Ejemplo de uso
CALL	Dir	Mem[SP] = PC + 1, SP- -, PC = Dir		CALL func
RET		SP++		-
		PC = Mem[SP]		-
PUSH	A	Mem[SP] = A, SP- -		-
PUSH	B	Mem[SP] = B, SP- -		-
POP	A	SP++		-
		A = Mem[SP]		-
POP	B	SP++		-
		B = Mem[SP]		-

Imagen Referencia: https://github.com/IIC2343/Syllabus-anteriores/blob/main/Apuntes%20(viejos)/5%20-%20Saltos%20y%20Subrutinas.pdf

Subrutina: Ejercicio I2 - 2012

a) Escriba en el assembly del computador básico un programa que calcule la división entre 2 números enteros, a y b, mediante la llamada a la subrutina $q = div(a, b)$, que también debe ser implementada. Cualquier detalle de implementación o aspecto que se asuma debe quedar claramente explicado. **(2 ptos.)**

Solución: Una implementación en Java de la división entera es la siguiente:

```
public byte div(byte a, byte b){
    byte res = 0;
    a = a-b;
    while (a >= 0)
    {
        res = res + 1;
        a = a-b;
    }
    return res;}
```

Imagen Referencia: https://github.com/IIC2343/Syllabus-anteriores/blob/main/Apuntes%20(viejos)/5%20-%20Saltos%20y%20Subrutinas.pdf

Subrutina: Ejercicio I1 - 2024-2

(a) (1.5 ptos.) Indique los valores de los registros `A` y `B` al finalizar la ejecución del código **(a)**.

(b) (1.5 ptos.) Indique los valores de los registros `A` y `B` al finalizar la ejecución del código **(b)**.

(c) (3 ptos.) El código **(c)** *debería* computar la multiplicación entre las variables `X` e `Y`, pero cuenta con errores que generan un resultado erróneo. Indique el valor de los registros `A`, `B` y la variable `res` al finalizar la ejecución del código. Luego, señale dónde se encuentra el o los errores y cómo se resuelven.

Subrutina: Ejercicio I1 - 2024-2

(a)

```
DATA:
res     0

CODE:
JMP _start

func_1:
SHR A, A
JCR func_2
CMP A, 0
JNE func_1
RET

func_2:
NOT B, A
MOV (B), 14
INC (res)
JMP func_1

_start:
MOV A, 6
CALL func_1
MOV B, (res)
```

(b)

```
DATA:
var   10

CODE:
JMP _start

func_1:
MOV B, (var)
NOT A, A
ADD A, 1
ADD B, A
RET

_start:
MOV A, 3
CALL func_1

end:
MOV A, (255)
```

(c)

```
DATA:
res     0
X       3
Y       4
iter    0

CODE:
MOV A, (Y)
PUSH A
CALL mult
MOV A, (res)
JMP end

mult:
POP B
sumar:
MOV A, (res)
ADD A, (X)
MOV (res), A
INC (iter)
MOV A, (iter)
CMP A, B
JNE sumar
RET

end:
```

Subrutina: Ejercicio I1 - 2024-2

Pregunta 4: Computador básico (6 ptos.)

Asuma que se encuentra trabajando como programador(a) en la oficina del famoso juego desarrollado completamente en Assembly: *Rollercoaster Tycoon*. Luego de unas horas, se da cuenta que la letra `T` del teclado de su computador ha dejado de funcionar, la que es de suma importancia al momento de escribir subrutinas por el uso de la instrucción `RET`. Adicionalmente, no cuenta con la opción de copiar y pegar texto, por lo que deberá desistir de su uso. Por el motivo anterior, deberá diseñar tres nuevas instrucciones que le permitan simular parte del comportamiento de la instrucción `RET`.

1. `MOV B, SP`: Almacena el valor del registro `SP` en el registro `B`.

Floats

Racionales: Single Precision Floating Point - IEEE754

- Un número de punto flotante, en estándar IEEE754, se ve así:

$$N = (-1)^{signo} \times 1.^{significante} \times 10^{(exponente - 127)_b}$$

```
Ejemplo:

 0.00101b = (1.01 * 10^-11)b

exponente - 127 = -3 ⇒ exponente = 124 = (01111100)

Según IEEE754: 0.00101b = 00111110001000000000000000000000
```

Racionales: Algoritmo de Suma

1. **Comparar los exponentes.** Hacer **shift right** a la mantisa del **número más pequeño** hasta que su exponente sea igual al del número más grande.
2. **Sumar** las mantisas.
3. **Normalizar la suma**. Esto puede ser haciendo shift right e incrementando el exponente, o shift left y decrementar el exponente. Depende del caso.
4. **Redondear el significante** al número apropiado de bits. Revisar nuevamente si está normalizado, en caso que no lo esté, volver al paso anterior.

Ejercicio:

Sumar $A = 1,11 \times 10^{11}$ y $B = 1,11 \times 10^{1}$

Racionales: Algoritmo de Suma

Sumar $A = 1,11 \times 10^{11}$ y $B = 1,11 \times 10^{1}$

Si sus exponentes son distintos, tome el número de menor

exponente y realice lo siguiente:

- Haga un shift right sobre los bits de significante.
- Incremente en una unidad los bits de exponente.
- Repita este procedimiento hasta que los exponentes de ambos números sean iguales

$A = 1,11 \times 10^{11}$ y $B = 0,0111 \times 10^{11}$

Racionales: Algoritmo de Suma

Sumar $A = 1,11 \times 10^{11}$ y $B = 0,0111 \times 10^{11}$

Sume los significantes, manteniendo el exponente

$$1,11 \times 10^{11} + 0,0111 \times 10^{11} = 10,0011 \times 10^{11}$$

Normalizar el resultado:

$$10,0011 \times 10^{11} = 1,00011 \times 10^{100}$$

Racionales: Algoritmo de Multiplicación

1. **Sumar los exponentes desfasados** y restar el desfase a la suma para obtener el nuevo exponente desfasado.
2. **Multiplicar** las mantisas.
3. **Normalizar el producto.** Esto puede ser haciendo shift right e incrementando el exponente, o shift left y decrementar el exponente. Depende del caso.
4. **Redondear el significante** al número apropiado de bits. Revisar nuevamente si está normalizado, en caso que no lo esté, volver al paso anterior.
5. **Verificar el signo.** Si los signos de ambos números es el mismo, es positivo. Si los signos son distintos, el producto es negativo. Para asignar el nuevo signo debemos hacer un **XOR** de los signos originales.

Ejercicio:

Multiplicar $C = 1{,}11 \times 10^{11}$ y $D = 1{,}11 \times 10^{1}$

Racionales: Algoritmo de Suma

Multiplicar $A = 1,11 \times 10^{11}$ y $B = 1,11 \times 10^{1}$

Sumar exponentes: 11 +1 = 100

Multiplicar mantizas:

1 ,11 × 1, 11

---

1 1 1

\+ 1 1 1

1, 0 1 0 1

¿Dudas?

Resumen Instrucciones: RISC-V

- **copiar_registro:** add t0, zero, t1
- **cargar_literal:** addi t0, zero, 6
- **Leer_memoria_directo:** lw t0, dir
- **Leer_memoria_indirecto:** lw t0, (t1)
- **Branch_if_equal:** beq t0, t1, dir
- **Branch_if_not_equal:** bne t0, t1, dir
- **Call_por_label:** jal ra, label
- **Return:** jalr zero, 0(ra)

Imagen Referencia: https://github.com/IIC2343/Syllabus-anteriores/blob/main/Apuntes%20(viejos)/8%20-%20Comunicaci%C3%B3n%20de%20CPU%20y%20Memoria%20con%20IO.pdf

Resumen Convención Llamada: RISC-V

- **Se deben usar los nombres definidos de los registros (zero, ra, t0, ...) y no su nombre según (x0, x1, x5, ...).**
- **Se deben usar los registros según su función:**
- **El registro ra:** Se usa para almacenar la dirección de retorno.
- **Los registros a0-a7:** Estos registros se utilizan para pasar argumentos a una subrutina.
- **Los registros a0-a1:** Se utilizan para almacenar valores de retornos de una subrutina.
- **El registro sp:** Se usa para indicar el tope del stack
- Los registros que deben ser respaldados por la subrutina **(callee) son: sp, s0-s11.** Los registros que deben ser respaldados por quien llama a la subrutina **(caller) son: ra, t0-t6 , a0-t7**

I/O

Arquitectura de Computadores: Dispositivos I/O

- Todos los dispositivos que no sean CPU o memoria, y se comuniqu con ellos, son llamados dispositivos de I/O

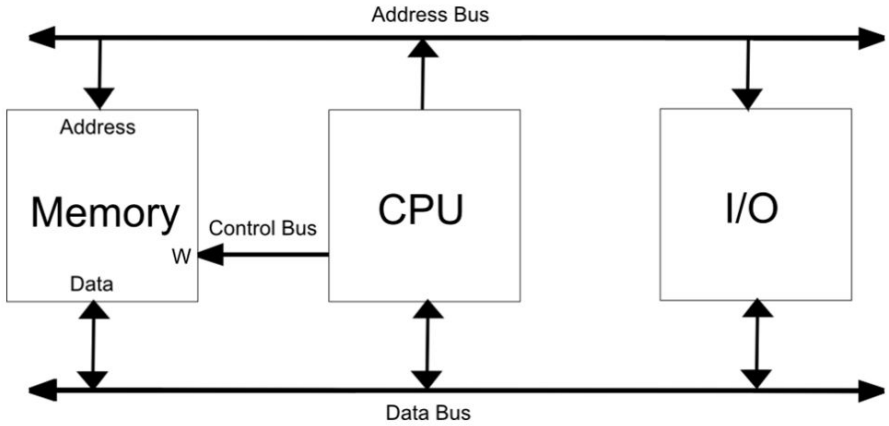

Imagen Referencia: https://github.com/IIC2343/Syllabus-anteriores/blob/main/Apuntes%20(viejos)/8%20-%20Comunicaci%C3%B3n%20de%20CPU%20y%20Memoria%20con%20IO.pdf

Arquitectura de Computadores: Dispositivos I/O

- Un dispositivo de I/O tiene un controlador encargado de comunicarse con la CPU y la memoria, y de controlar la parte electromecánica. Todos los dispositivos que no sean CPU o memoria, y se comuniquen con ellos, son llamados dispositivos de I/O

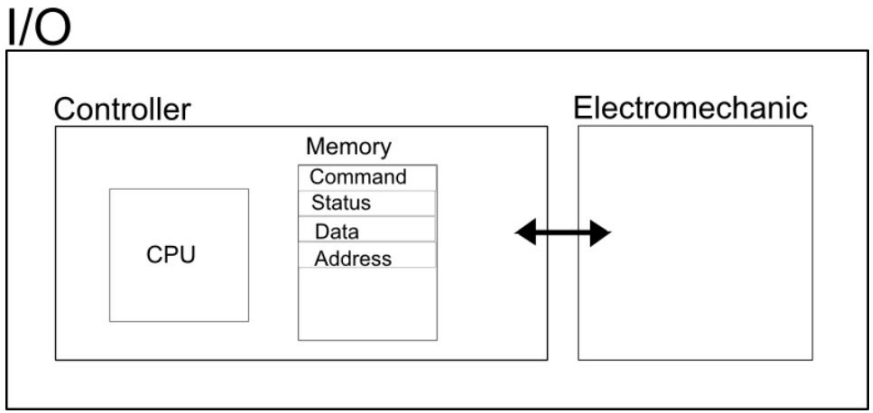

Imagen Referencia: https://github.com/IIC2343/Syllabus-anteriores/blob/main/Apuntes%20(viejos)/8%20-%20Comunicaci%C3%B3n%20de%20CPU%20y%20Memoria%20con%20IO.pdf

Arquitectura de Computadores: Memory Map

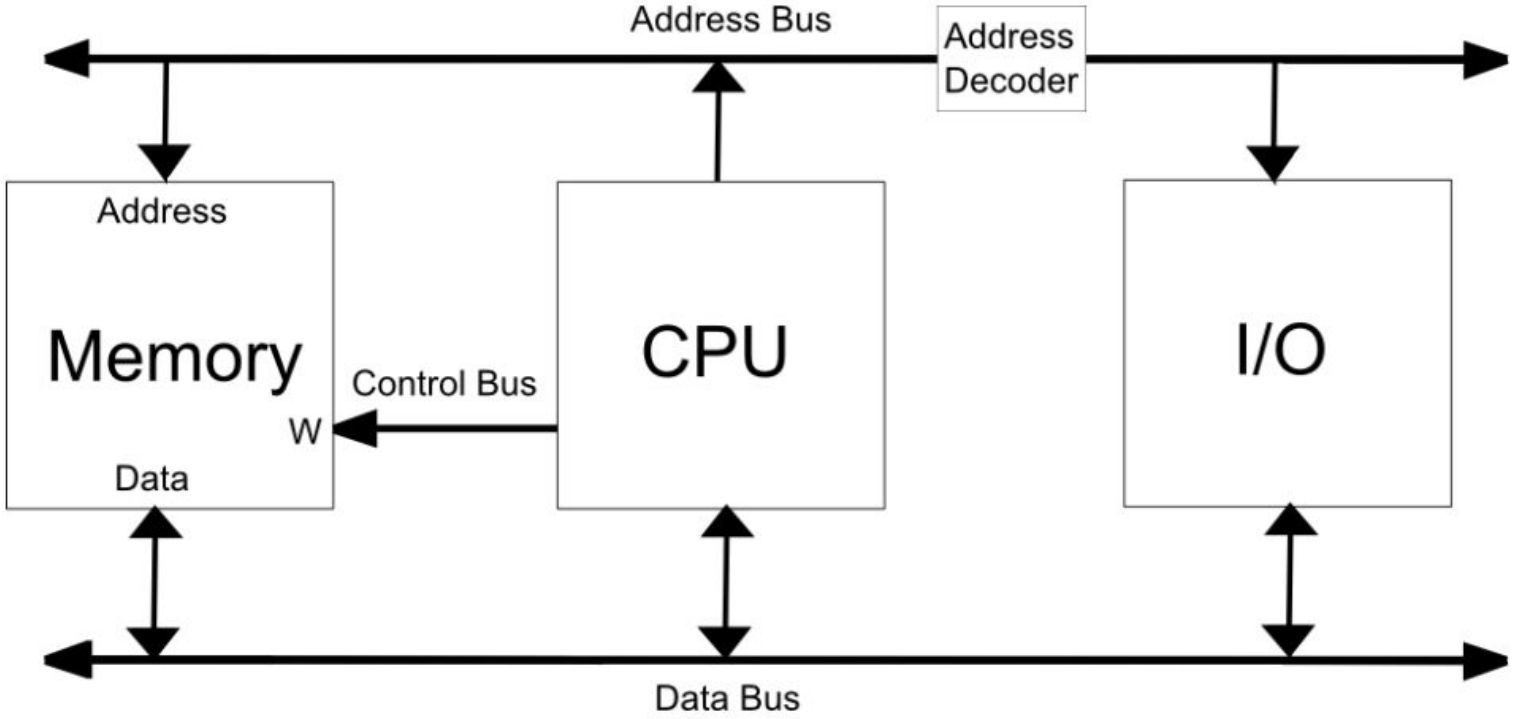

Imagen Referencia: https://github.com/IIC2343/Syllabus-anteriores/blob/main/Apuntes%20(viejos)/8%20-%20Comunicaci%C3%B3n%20de%20CPU%20y%20Memoria%20con%20IO.pdf

Arquitectura de Computadores: Port I/O

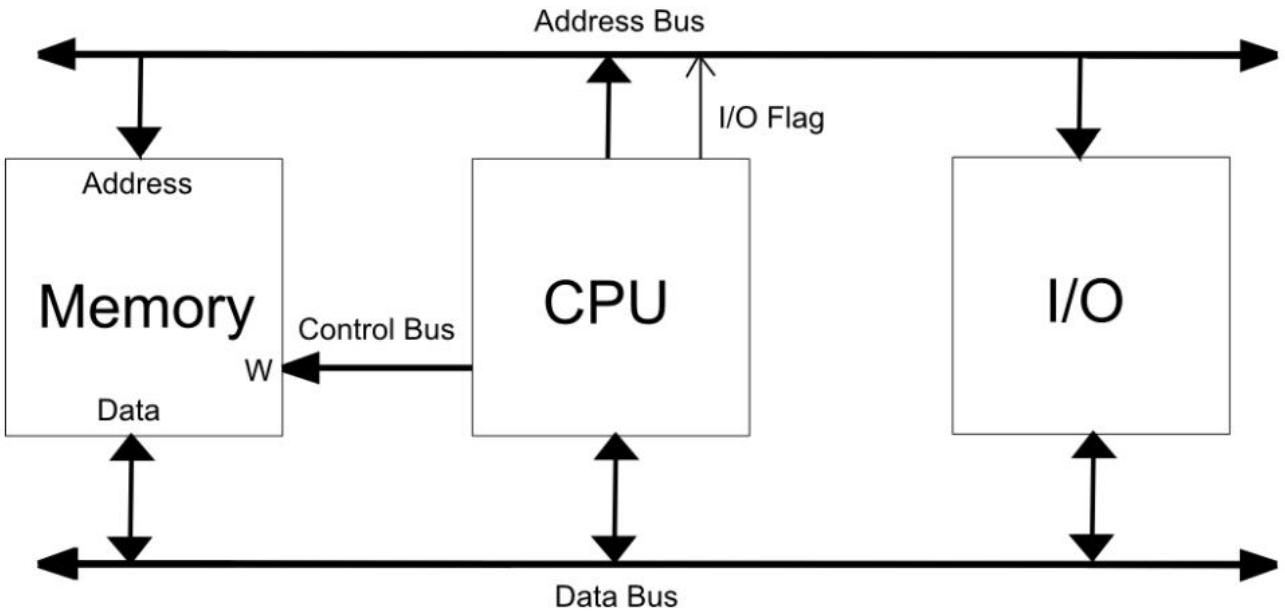

Imagen Referencia: https://github.com/IIC2343/Syllabus-anteriores/blob/main/Apuntes%20(viejos)/8%20-%20Comunicaci%C3%B3n%20de%20CPU%20y%20Memoria%20con%20IO.pdf

¿Dudas?

Revisaremos tres modelos de interacción entre alumnos y profesor

Sin proyector ni pizarrón ni copias de las guías, alumnos tímidos:

1. **Polling**

Sin proyector ni pizarrón ni copias de las guías, alumnos

preguntones y participativos:

2. **Interrupción**

Pizarra interactiva, alumnos preguntones:

3. **Interrupción con acceso directo a memoria**

Imagen Referencia: https://github.com/IIC2343/Syllabus-anteriores/blob/main/Apuntes%20(viejos)/8%20-%20Comunicaci%C3%B3n%20de%20CPU%20y%20Memoria%20con%20IO.pdf

Arquitectura de Computadores: Polling

- ¿Necesitamos cambio de hardware?
- NO

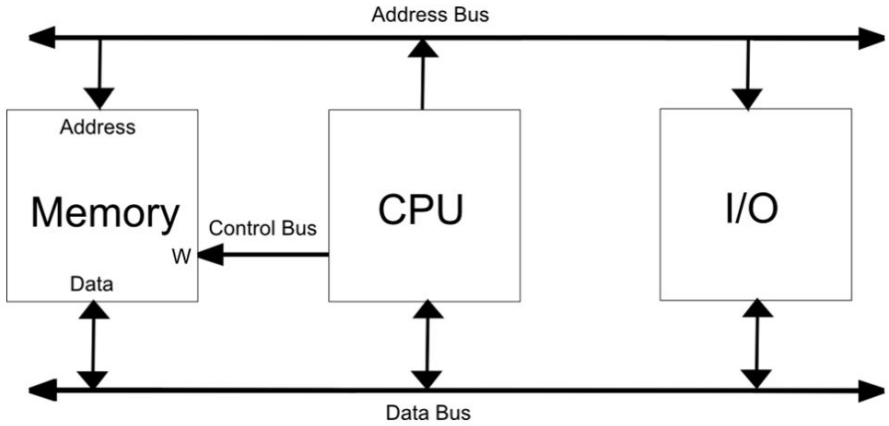

Imagen Referencia: https://github.com/IIC2343/Syllabus-anteriores/blob/main/Apuntes%20(viejos)/8%20-%20Comunicaci%C3%B3n%20de%20CPU%20y%20Memoria%20con%20IO.pdf

Arquitectura de Computadores: Interrupción

- ¿Necesitamos cambio de hardware?
- SI, una señal de **solicitud de interrupción**

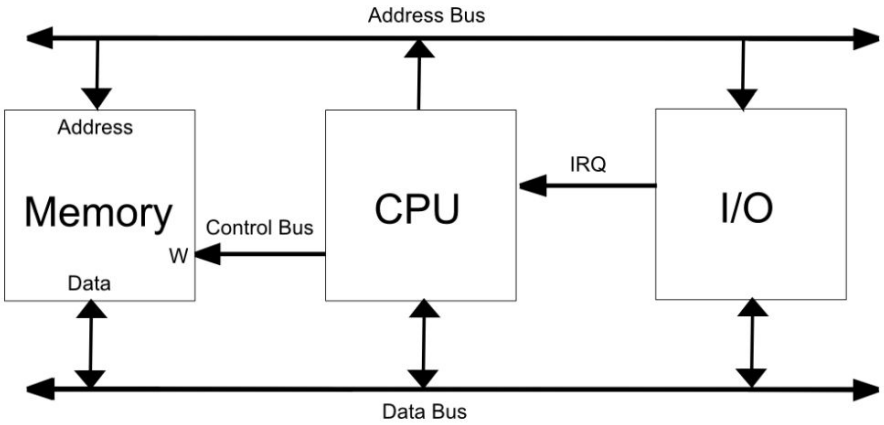

Imagen Referencia: https://github.com/IIC2343/Syllabus-anteriores/blob/main/Apuntes%20(viejos)/8%20-%20Comunicaci%C3%B3n%20de%20CPU%20y%20Memoria%20con%20IO.pdf

Arquitectura de Computadores: Interrupción

- Al usar varios dispositivos, es necesario incorporar un controlador de interrupciones y una tabla de vectores de interrupción

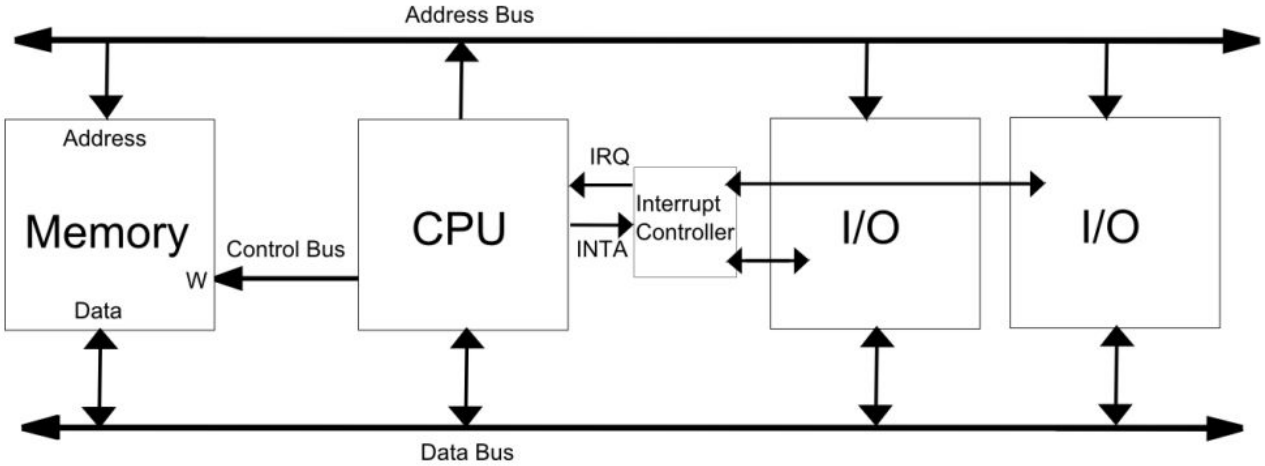

Imagen Referencia: https://github.com/IIC2343/Syllabus-anteriores/blob/main/Apuntes%20(viejos)/8%20-%20Comunicaci%C3%B3n%20de%20CPU%20y%20Memoria%20con%20IO.pdf

Arquitectura de Computadores: DMA

- En general, un controlador de DMA tendrá al menos los siguientes componentes:
  - Registro de comandos y estado
  - Registros de dirección de origen y destino
  - Registro contador de palabras
  - Buffer de almacenamiento temporal
  - Unidad de control

Imagen Referencia: https://github.com/IIC2343/Syllabus-anteriores/blob/main/Apuntes%20(viejos)/8%20-%20Comunicaci%C3%B3n%20de%20CPU%20y%20Memoria%20con%20IO.pdf

¿Dudas?

Arquitectura de Computadores: Ejercicio I2-2011

- ¿Qué modelo de interacción entre CPU, memoria e I/O recomendaría y por qué, para un disco duro, un mouse y una cámara de video?

Arquitectura de Computadores: Ejercicio I2-2011

- **¿Qué modelo de interacción entre CPU, memoria e I/O recomendaría y por qué, para un disco duro, un mouse y una cámara de video?**

  Para un disco duro se recomienda el esquema de interrupciones con DMA, ya que es un dispositivo con alto nivel de interacción con la CPU y sus transferencias son generalmente de gran tamaño.

  Para un mouse se recomienda el esquema de interrupciones sin DMA, ya que, a pesar de que presenta un alto nivel de interacción, las transferencias son pequeñas.

  Finalmente, para la cámara de video, se recomienda el esquema de interrupciones con DMA, ya que el tamaño de las transferencias es muy alto y con bastante frecuencia.

Arquitectura de Computadores: Ejercicio I2-2018

- Al implementar memory-mapped I/O en el computador básico, ¿qué buses se deben intervenir y por qué?

Arquitectura de Computadores: Ejercicio I2-2018

- **Al implementar memory-mapped I/O en el computador básico, ¿qué buses se deben intervenir y por qué?**

  En principio, se deben intervenir los buses de dirección (Memory Address), de control (W) y de datos (Data In y Data Out) de la memoria. Esto se debe a que entre todos ellos se instalará un address decoder, que, dependiendo de la dirección entrante por el bus de direcciones y del estado del bit W, decidirá si dirigir la señal y los datos hacia la RAM o hacia algún dispositivo de I/O. Esta intervención se realiza a este nivel porque permite que el resto de la microarquitectura no requiera modificaciones para funcionar con este esquema de I/O.

Clase 17 - Repaso I2

IIC2343 - Arquitectura de Computadores

Profesor:
- Felipe Valenzuela González
Correo:
frvalenzuela@alumni.uc.cl