Departament d'Enginyeria Informàtica i Matemàtiques — Universitat Rovira i Virgili	Estructura de Computadores
	EC
	Curso 17/18
	Primera Convocatoria
	Práctica 2: Procesador MIPS Monociclo

Procesador MIPS Monociclo

Comentarios

• La práctica se realizará **en GRUPOS DE 2 PERSONAS**

• Se realizará una entrevista y prueba del funcionamiento de los circuitos con todos los integrantes del grupo en la sesión de laboratorio que tienen asignada.

• De entre todas las prácticas del curso que demuestren **un funcionamiento correcto y completo**, y que en **la evaluación normal de la asignatura** hayan obtenido una media **igual o superior a 5**, obtendrán un punto adicional a sumar a la NOTA FINAL de la asignatura:

- La implementación que utilice un coste hardware menor (**estimated area** en TkGate)
- La implementación con un tiempo de ciclo menor. IMPORTANTE: no se permite modificar los tiempos de retardo que incorporan por defecto los elementos del simulador

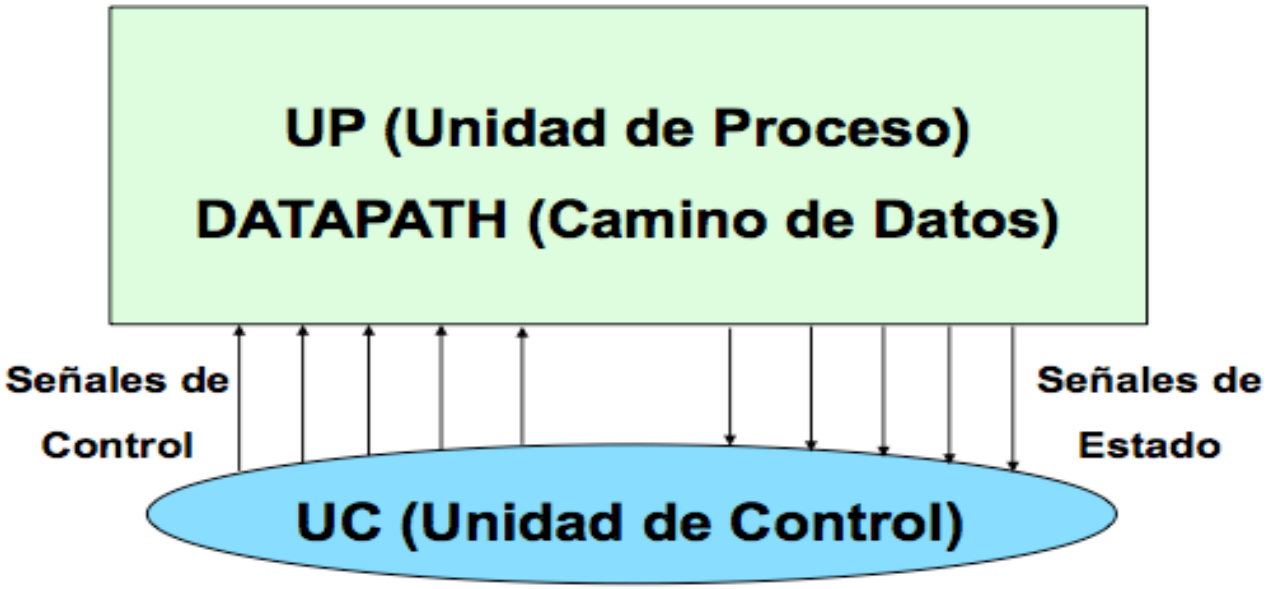
En el caso de empate entre varios grupos, el punto se repartirá a partes iguales entre todos ellos.

• El informe (**obligatoriamente en PDF**) y los ficheros TkGate con los circuitos implementados se subirán al moodle **en un único fichero ZIP**. Las entregas que no respeten el formato establecido se considerarán como **NO PRESENTADAS**.

Introducción

La práctica consiste en la implementación, mediante un simulador de circuitos digitales, de un procesador MIPS monociclo. Con el nombre de MIPS (siglas de *Microprocessor without Interlocked Pipeline Stages*) se conoce a toda una familia de microprocesadores de arquitectura RISC desarrollados por MIPS Technologies. La principal característica de un procesador monociclo es que ejecuta todas sus instrucciones de manera secuencial y con CPI=1.

Un procesador se divide en dos partes claramente diferenciadas: Unidad de Proceso (o Camino de Datos) y Unidad de Control. La unidad de proceso contiene todo el hardware necesario para la ejecución del repertorio de instrucciones del procesador. La unidad de control es la encargada de indicar a la unidad de proceso el modo en que se tiene que comportar. Para ello implementa una máquina de estados que, en función del estado actual y de unas señales de estado que llegan desde la unidad de proceso, genera unas señales de control para dirigirla. A continuación se presenta una figura que resume este comportamiento.



En esta práctica se diseñarán ambas unidades (Unidad de Proceso y Unidad de Control) partiendo del diseño parcial que se proporcionan en los laboratorios. Por simplicidad, de todo el repertorio de instrucciones que contempla el ISA del procesador MIPS, inicialmente se tratarán el siguiente subconjunto básico:

1. **Instrucciones aritmético-lógicas**

- **add** rd, rs, rt — #rd = rs + rt — *add $t0, $a0, $a1*
- **sub** rd, rs, rt — #rd = rs - rt — *sub $t0, $a0, $a1*
- **and** rd, rs, rt — #rd = rs & rt — *and $t0, $a0, $a1*
- **or** rd, rs, rt — #rd = rs | rt — *or $t0, $a0, $a1*
- **slt** rd, rs, rt — #si (rs < rt) rd = 1 #sino rd = 0 — *slt $t0, $a0, $a1*

2. **Instrucciones de referencia a memoria**

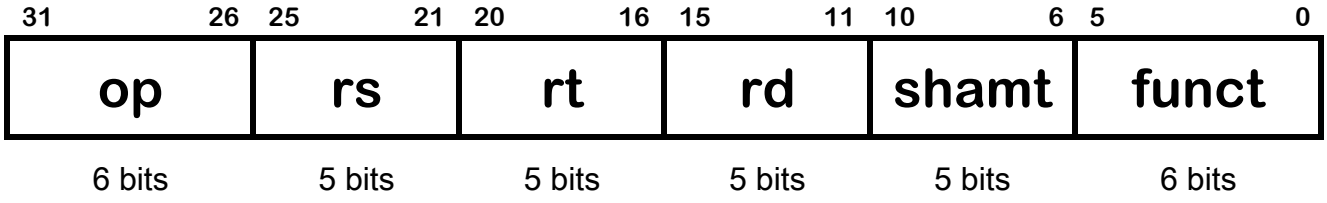
- **lw** rt, dir(rs) — #rt = Memoria[rs+dir] — *lw $s0, 100($a1)*
- **sw** rt, dir(rs) — #Memoria[rs+dir] = rt — *sw $s0, 100($a1)*

3. **Instrucciones de control de flujo**

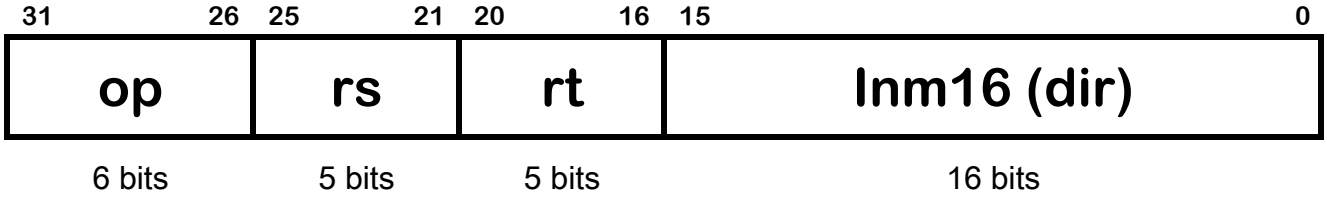
- **beq** rs, rt, dir — #branch on equal (saltar si es igual) #si (rs = rt) $pc = $pc + dir — *beq $t0, $t1, etiqueta*
- **j** dir — #jump (saltar) #$pc = dir — *j etiqueta*

El formato de cada una de las instrucciones anteriores es el siguiente:

1. **Formato R (Register):** `add, sub, and, or, slt`



2. **Formato I (Inmediate):** `lw, sw, addi, beq`



3. **Formato J (Jump):** `j`

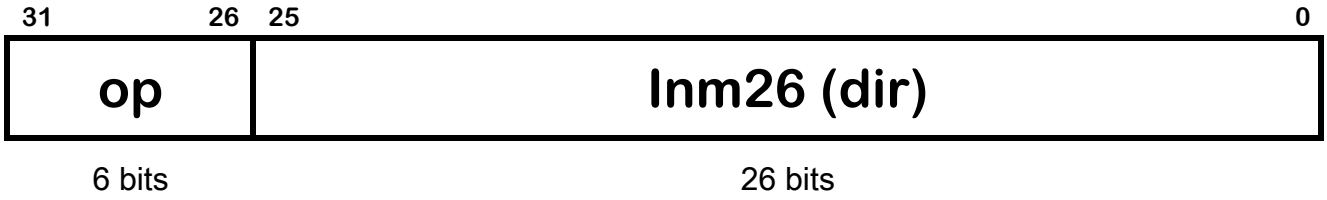


Figura 1. Formato de las instrucciones del procesador MIPS.

Se pueden encontrar detalles del comportamiento del procesador MIPS monociclo en las transparencias del curso, en documentos del moodle o bien en el capítulo 4 del siguiente libro:

> David A. Hennessy & John L. Patterson, **Estructura y Diseño de Computadores: La Interfaz Hardware / Software,** Editorial Reverte, 4ª Edición, 2011.

Para la implementación, se utilizará como herramienta de soporte la aplicación TkGate. Esta herramienta consta de un editor gráfico, un simulador de circuitos digitales desarrollado con Tcl/TK y una librería de funciones que incorpora desde componentes básicos como puertas lógicas hasta componentes más complejos como registros y memoria.

De cara a un mejor seguimiento de la práctica, se han divido las distintas tareas a realizar en las 5 fases que se detallan a continuación:

FASE 1: Fetch y Read

TAREA 1. Realizad la parte del circuito del procesador que se encarga del ***fetch*** de las instrucciones y que se muestra en la siguiente figura.

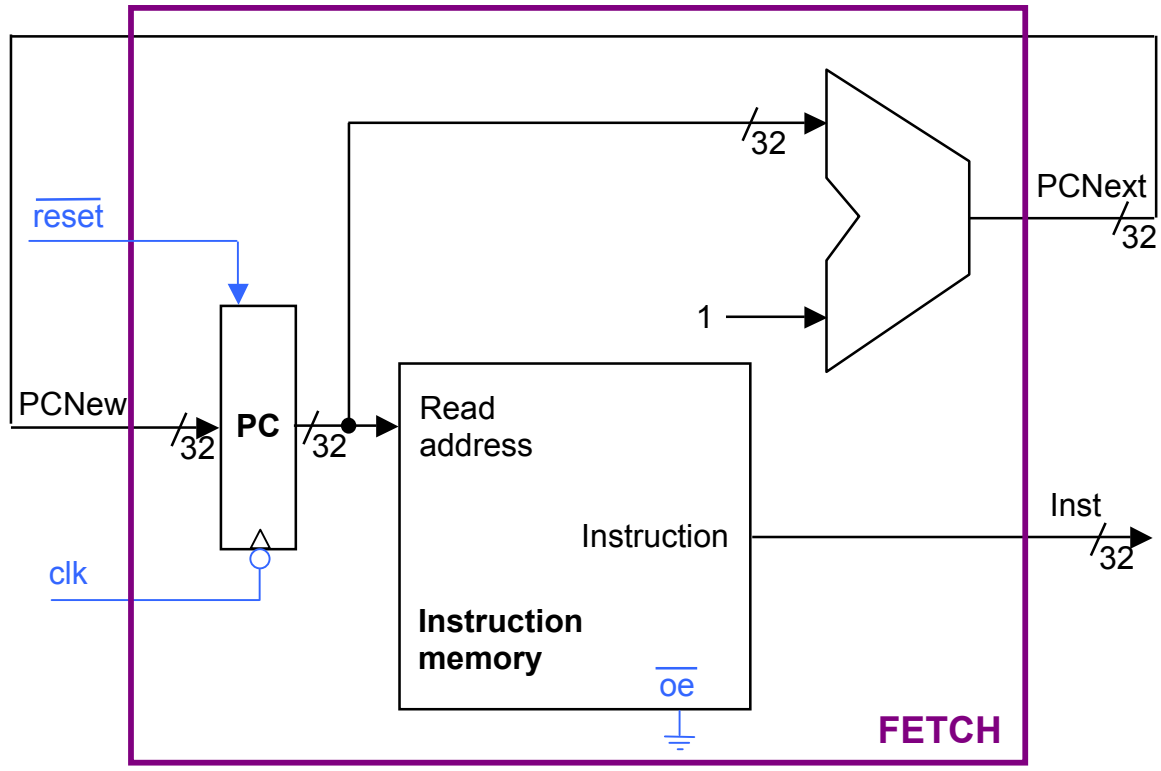


Figura 2. Etapa FETCH del procesador monociclo.

OBSERVACIONES:

- El tamaño de las direcciones del procesador es de 32 bits.
- El tamaño de las instrucciones del ISA es de 32 bits.
- Es necesario utilizar una señal de reloj global que active la carga del registro PC.
- Optativamente se puede considerar un interruptor global (***#reset***) que limpie el valor del PC (***#clr***).
- En el diseño original, la memoria está organizada en palabras de 1 byte, por lo tanto, para pasar a la siguiente instrucción es necesario sumar 4 al PC. Para simplificar el diseño utilizaremos una memoria con un tamaño de palabra de 4 bytes, por lo tanto, para ir a la siguiente instrucción basta con sumar uno a la dirección.
- Sería conveniente realizar las pruebas necesarias para verificar el correcto funcionamiento del circuito incorporando, de forma temporal, interruptores, DIP, leds y visualizadores.

TAREA 2. Utilizando el ***banco de registros*** que se proporciona en la librería, probad el circuito que implementa la etapa de lectura de ***los registros*** y que se muestra en la siguiente figura. Realizad varios ciclos de escritura y lectura para comprobar el correcto funcionamiento del mismo.

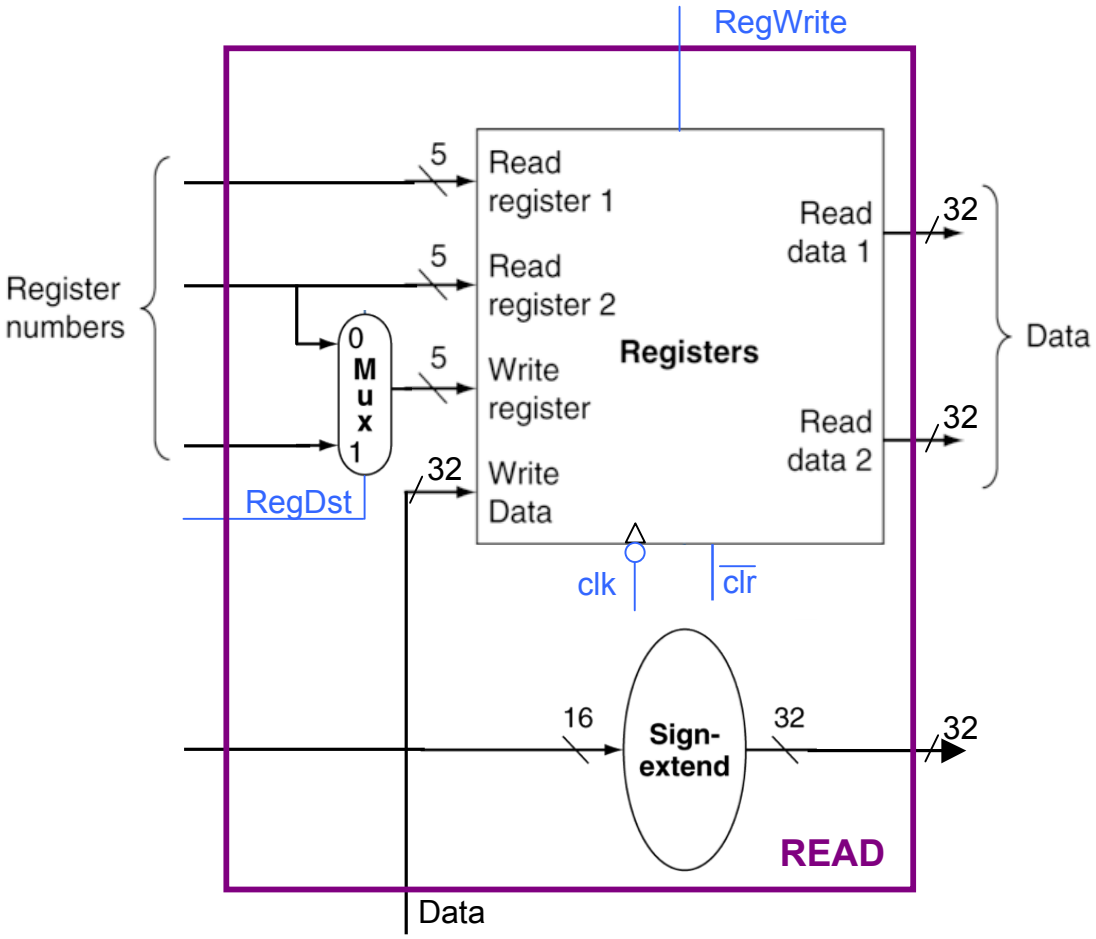


Figura 3. Banco de registros y extensión de signo del procesador monociclo MIPS.

OBSERVACIONES:

- El banco de registros es de 32 entradas (señales de 5 bits para seleccionar el registro).
- El interruptor global (***#reset***) limpia el valor de todos los registros del banco (esto implica añadir una nueva señal ***#clr*** al banco de registros).
- La señal de control (***RegWrite***) se activará a 1 cuando se quiera realizar la escritura de un registro. Independientemente del valor de la señal ***RegWrite***, la lectura de registros siempre está activa.
- Sería conveniente realizar las pruebas necesarias para verificar el correcto funcionamiento del circuito incorporando, de forma temporal, interruptores, DIP, leds y visualizadores.

FASE 2: Execute y Mem

TAREA 3. Utilizando el ***banco de registros*** que se proporciona en la librería , realizad la parte del circuito del procesador que se encarga de la ejecución de las instrucciones de **aritmético-lógicas (Formato R)** y que se muestra en la siguiente figura.

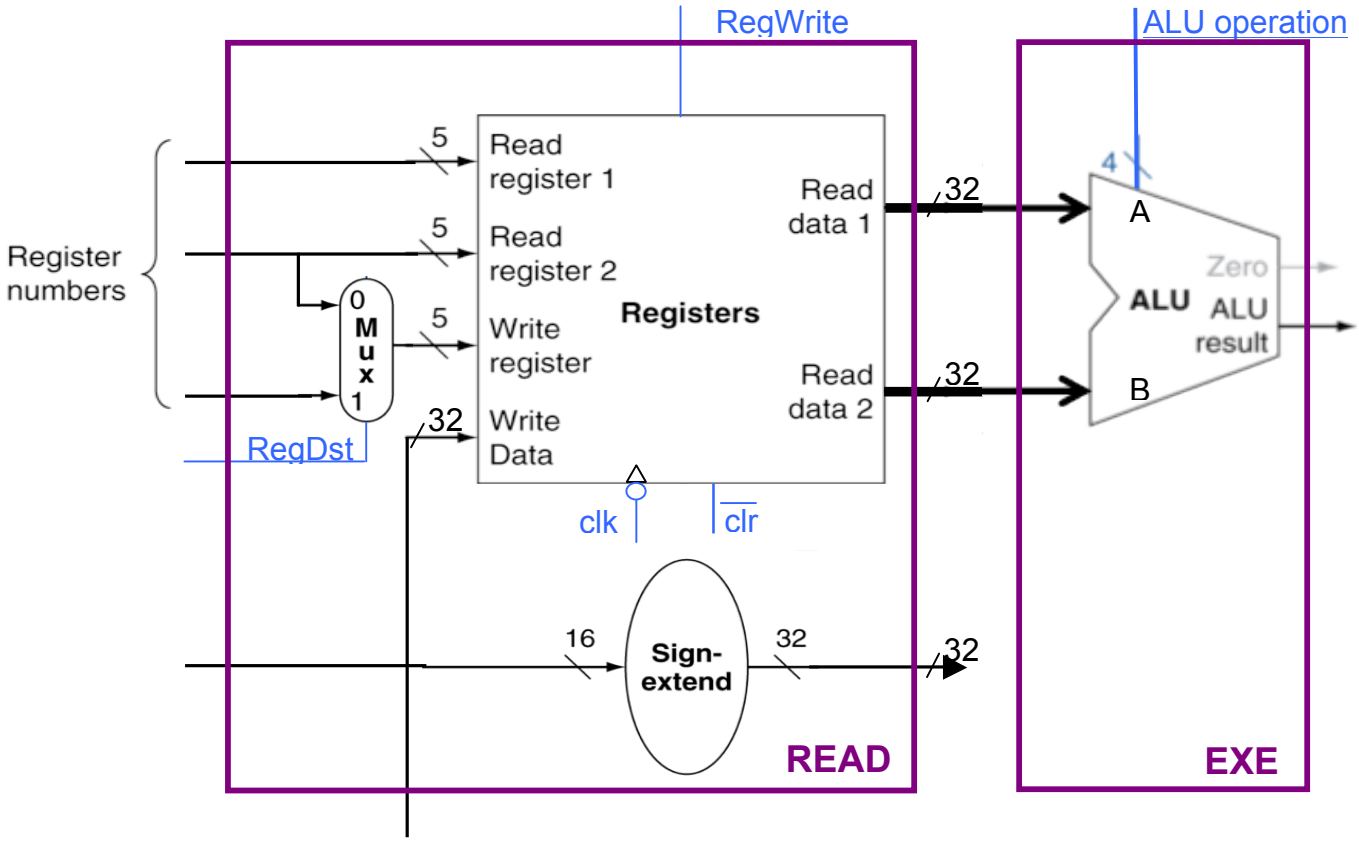


Figura 4. Etapa de ejecución de las instrucciones tipo R.

OBSERVACIONES:

- La ALU realiza las operaciones de suma, resta, Y lógico, O lógico y "activar si menor que" (ver instrucciones aritmético-lógicas) y muestra el resultado por la salida ***ALU Result***
- A continuación se muestran los valores de la señal de control ***ALU operation***

ALU operation	ALU Function	Zero
0000	A **AND** B	✓
0001	A **OR** B	✓
0010	A + B	✓
0110	A - B	✓
0111	si (A < B) ALUresult = 1 sino ALUresult = 0	✓

Tabla 1. Relación entre las señales de control de la ALU y la operación de la ALU correspondiente.

- La ALU también dispone de la señal de salida ***Zero*** que se activa a 1 siempre y cuando el resultado de la operación realizada sea igual a cero.
- El banco de registros recibe los números de registros de lectura y escritura de la propia instrucción.
- Sería conveniente realizar las pruebas necesarias para verificar el correcto funcionamiento del circuito incorporando, de forma temporal, interruptores, DIP, leds y visualizadores.

TAREA 4. Realizad la parte del circuito del procesador que se encarga de la ejecución de las instrucciones básicas de **formato I y J (beq** y **j)** y que se muestra en la siguiente figura.

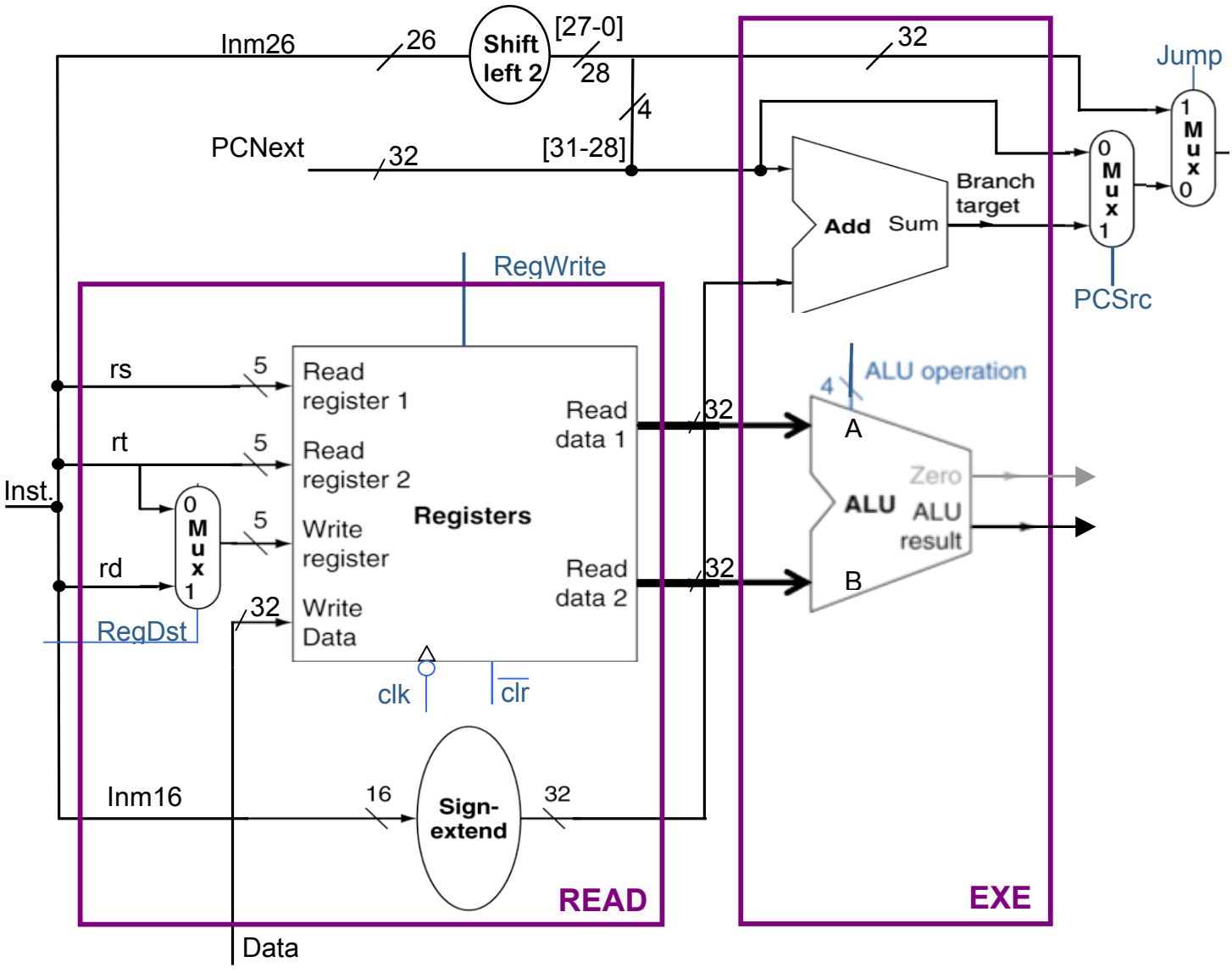


Figura 5. Etapa de ejecución de las instrucciones de salto beq (tipo I) y j (tipo J).

OBSERVACIONES:

- Es necesario realizar las pruebas necesarias para verificar el correcto funcionamiento del circuito incorporando, de forma temporal, interruptores, DIP, leds y visualizadores.

TAREA 5. A partir del diseño de la tarea anterior, añadir la parte del circuito del procesador que se encarga de la ejecución de las instrucciones de acceso a memoria, **lw y sw (Formato I)** y que se muestra en la siguiente figura.

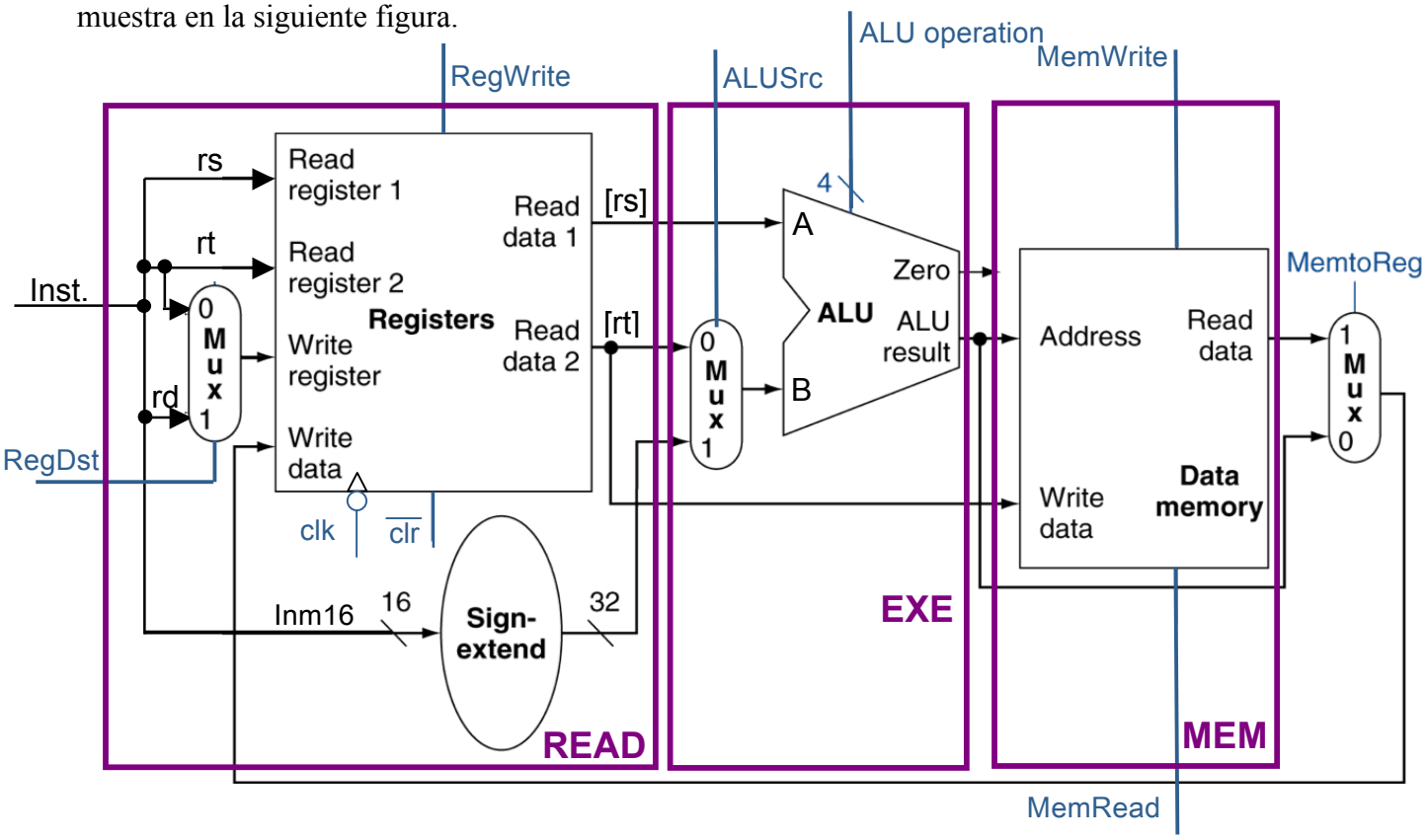


Figura 6. Etapa de ejecución de las instrucciones tipo I de acceso a memoria: lw y sw.

OBSERVACIONES:

- La extensión de signo se realiza a partir de los bits de menor peso codificados en la instrucción (Inm16).
- La señal de control ***ALUSrc*** se encarga de seleccionar la fuente del 2º operando que utiliza la ALU: el operando puede venir del banco de registros (instrucciones de formato R) o bien puede estar codificado en la propia instrucción (instrucciones de formato I).
- Las señales de control ***MemRead*** y ***MemWrite*** se encargan de activar la lectura/escritura en la memoria de datos, según se trate de una instrucción de carga (lw) o almacenamiento (sw).
- La señal de control ***MemtoReg*** selecciona el origen del dato que se escribirá en el banco de registros: (1) de la ALU para el caso de las instrucciones tipo R o (2) de la memoria para el caso de las instrucciones tipo I.
- La señal de control ***RegDst*** selecciona el registro de escritura del banco de registros dependiendo del tipo de instrucción: (1) **rd** para las de formato R o **rt** para las de formato I.
- Sería conveniente realizar las pruebas necesarias para verificar el correcto funcionamiento del circuito incorporando, de forma temporal, interruptores, DIP, leds y visualizadores.

FASE 3: Unidad de Proceso y Unidad de Control

TAREA 6. Unid todos los componentes implementados hasta el momento para obtener la **Unidad de Proceso** completa del procesador MIPS monociclo. En la siguiente figura se muestra el esquema simplificado.

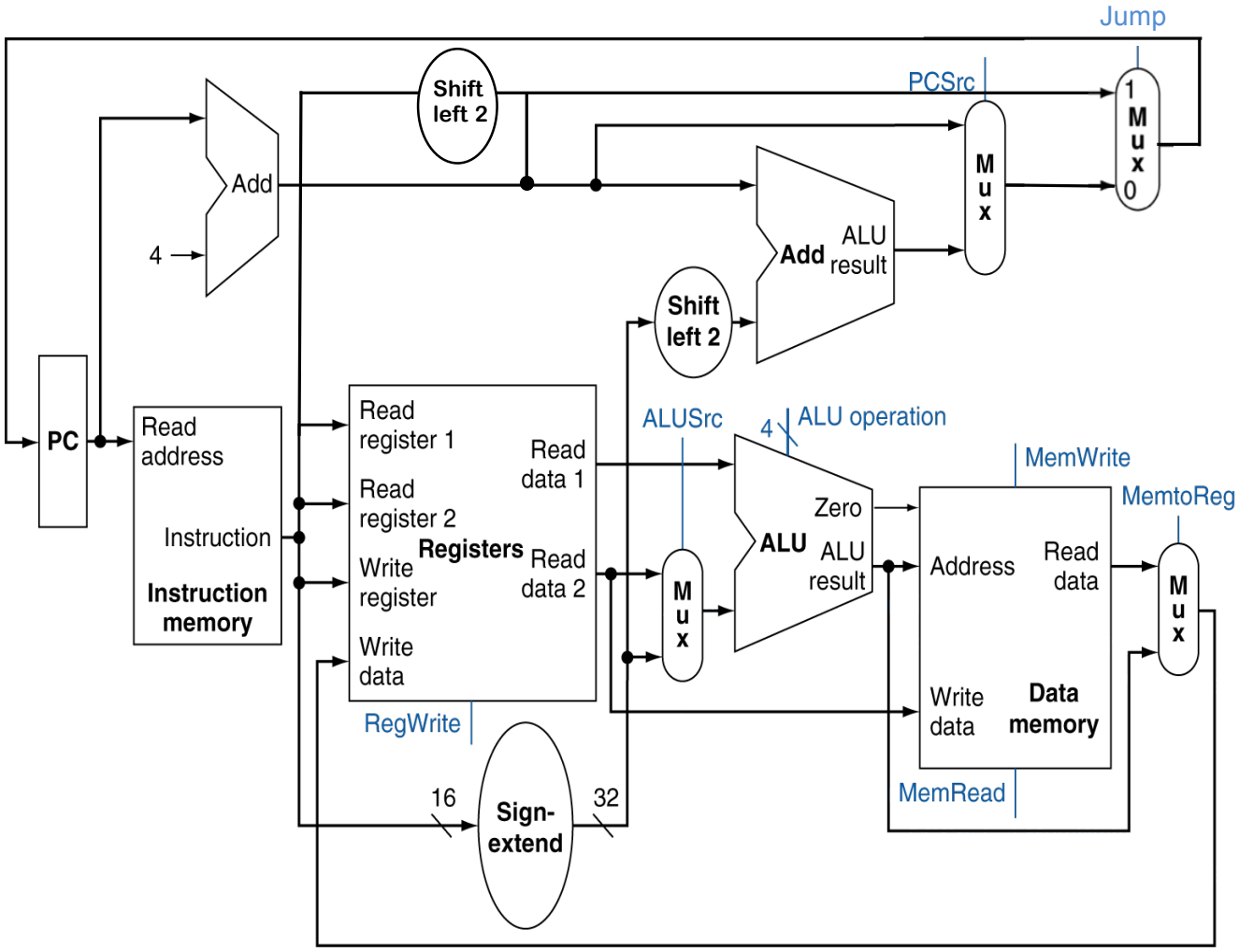


Figura 7. Diseño completo simplificado de la Unidad de Proceso básica del procesador MIPS.

OBSERVACIONES:

- Sería conveniente realizar las pruebas necesarias para verificar el correcto funcionamiento del circuito incorporando, de forma temporal, interruptores, DIP, leds y visualizadores.

TAREA 7. Realizad la parte del circuito que se encarga de la **Unidad de Control** del procesador MIPS y que se muestra junto a la Unidad de Proceso en la siguiente figura.

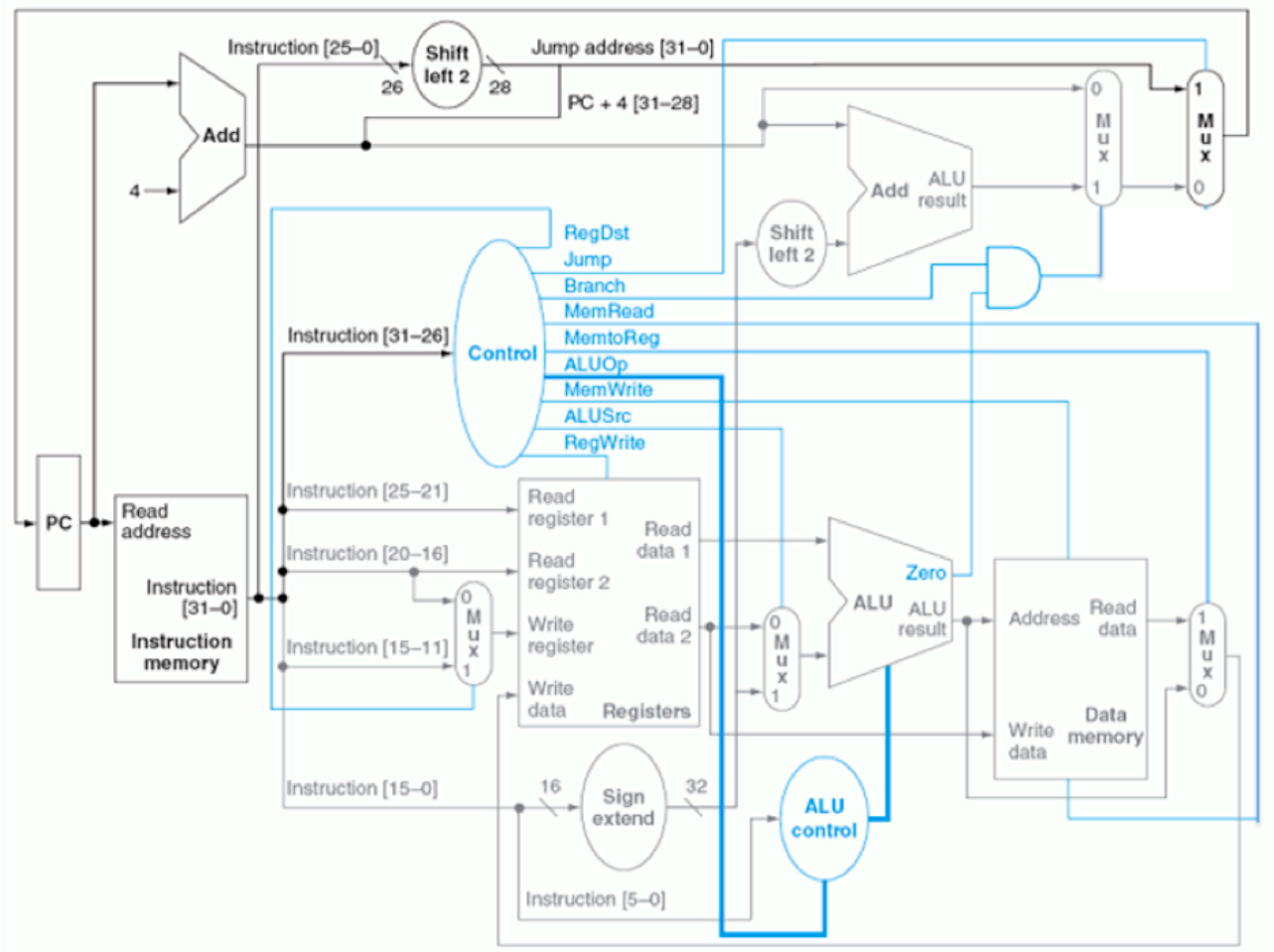


Figura 8. Diseño completo básico de la unidad de proceso y la unidad de control del procesador MIPS.

OBSERVACIONES:

- La señal ***PCSrc*** no es una señal propia de la unidad de control.
- En la figura se muestra la unidad de control separada en dos módulos: **control** y **ALU control**. Es decisión del alumno realizar el control en dos o un único módulo.
- Para la implementación de la Unidad de Control existen dos opciones: (1) cableada o (2) microprogramada. MUY IMPORTANTE: en esta práctica se asumirá una implementación microprogramada de la Unidad de Control.

- La unidad de control (1) recibe como entrada el **código de operación** y **función** de la instrucción y (2) activa/desactiva las señales de control. En la siguiente tabla se muestran los códigos de operación y función de las diversas instrucciones, en **hexadecimal**.

	Op	Func.
add	0	20 h
sub	0	22 h
and	0	24 h
or	0	25 h
slt	0	2A h
addi	8 h	-
lw	23 h	-
sw	2B h	-
beq	4 h	-
j	2 h	-

Tabla 2. Códigos de operación y función de las diversas instrucciones.

- A continuación se muestran los valores de las señales de control en función de la instrucción. Las columnas adicionales serán o no necesarias en función del diseño realizado por el alumno. Se pueden añadir todas las columnas adicionales que se necesiten.

Op	RegDst	Branch	Jump	MemRead	MemToReg	ALUCtrl	ALUOp	MemWrite	ALUSrc	RegWrite		
add	1	0	0	0	0	0010	10	0	0	1		
sub	1	0	0	0	0	0110	10	0	0	1		
and	1	0	0	0	0	0000	10	0	0	1		
or	1	0	0	0	0	0001	10	0	0	1		
slt	1	0	0	0	0	0111	10	0	0	1		
lw	0	0	0	1	1	0010	00	0	1	1		
sw	X	0	0	0	X	0010	00	1	1	0		
beq	X	1	0	0	X	0110	01	0	0	0		
j	X	0	1	0	X	XXXX	XX	0	0	0		

Tabla 3. Valor de las señales de control para cada instrucción.

FASE 4: Funcionalidades Adicionales

TAREA 8. Añadir los elementos necesarios, a la **Unidad de Proceso** y a la **Unidad de Control** anterior para poder implementar en el procesador MIPS monociclo la siguiente funcionalidad: Capacidad de poder leer y escribir en varias terminales TTY. Se usaran las siguientes instrucciones, con el formato:

```
lw $a, label($b)

si (label=0xFFFF) $a = PortTTY[$b] | RTS<<8

sw $a, label($b)

si (label=0xFFFF) PortTTY[$b] = $a   #8 bits
```

Donde el lw (load word) con label de valor 0xFFFF lee del terminal TTY[$b] un posible carácter ASCII (8 bits) que haya sido pulsado por el usuario; añade un 9 bit que será la señal RTS del TTY. La instrucción sw (store Word) con label de valor 0xFFFF escribirá en el terminal TTY[$b] el carácter ASCII (8 bits de menor peso) del registro $a. Consideraremos que hay un mínimo de 2 terminales y un máximo de 4 terminales TTY (o sea $b ira de 0 a 3)

OBSERVACIONES:

- Indicad las modificaciones realizadas a la unidad de proceso para implementar esta modificación de las instrucciones.
- Indicad los valores de las señales de control existentes así como de las nuevas señales que se pudieran añadir si fuera necesario.
- Sería conveniente realizar las pruebas necesarias para verificar el correcto funcionamiento del circuito incorporando, de forma temporal, interruptores, DIP, leds y visualizadores.
- Añadir un código ensamblador que muestre la transferencia de caracteres a y desde el TTY[0] hasta TTY[1]. Por ejemplo el código de una multiplicación de dos números que serán introducidos por el TTY[0] y se mostrara el resultado de la multiplicación per el TTY[1].

FASE 5: Prueba de Funcionamiento y Evaluación del Rendimiento

TAREA 9. Realizad una **prueba conjunta** de funcionamiento de las instrucciones del repertorio ISA **básico** que se ha asumido en esta práctica, a través de la ejecución del programa de prueba que se proporciona.

TAREA 10. Realizad un programa ensamblador de prueba propio y evaluad el **funcionamiento conjunto** de todas las instrucciones del repertorio ISA que se han asumido en esta práctica.