Trabajo Teórico-Práctico de Laboratorio Número 1

Arquitectura de Computadoras 22/04/2026

Instrucciones

- El laboratorio consta de 4 ejercicios teórico-prácticos y 6 ejercicios de laboratorio, algunos de los cuales no poseen una única solución sino múltiples soluciones posibles.
- Los ejercicios propuestos en el presente trabajo práctico pueden ser resueltos en forma individual o grupal.
- El planteo de la solución debe realizarse basándose en lo aprendido en las clases teóricas.
- Puede utilizar las clases de consulta para consultar ejercicios de enunciado y verificar la validez de la solución obtenida.
- Posteriormente, se tomará una evaluación con ejercicios similares a los de este práctico, la cual deberá ser resuelta en forma individual. La fecha de evaluación se encuentra en el sitio web de la materia.

Objetivo

El objetivo del laboratorio es familiarizar a los estudiantes con procesadores de arquitectura RISC-V utilizando el simulador Ripes, analizando el funcionamiento del camino de datos en procesadores de ciclo único y segmentados, y estudiando el impacto de los riesgos de datos en la performance.

Conceptos involucrados

- Procesador uniciclo: camino de datos, implementación del control con tabla de verdad.
- Procesador segmentado: camino de datos, registros de segmentación, implementación del control propagado.
- Agregado de instrucciones. Modificaciones al camino de datos y al control.
- Determinación de la duración del tiempo de ciclo en cada procesador. Camino crítico.
- Riesgos de datos: concepto, detección del riesgo, solución mediante adelantamiento.
- Influencia de los distintos tipos de riesgos en el CPI de un programa.
- Optimización de código mediante reordenamiento de instrucciones.

Problemas Teorico-Prácticos

1) Examinando la arquitectura del procesador uniciclo, cuya trayectoria de datos se adjunta al final de este práctico, y con la tabla de verdad del control del procesador uniciclo que se muestra a continuación, conteste las siguientes preguntas:

Señal	Tipo R	lw	sw	beq
FuenteALU	0	1	1	0
MemaReg	0	1	X	X
EscrReg	1	1	0	0
LeerMem	0	1	0	0
EscrMem	0	0	1	0
SaltoCond	0	0	0	1
ALUOP1	1	0	0	0
ALUOP0	0	0	0	1

   a) ¿Por qué el procesador uniciclo tiene dos memorias separadas y no una sola?
   b) ¿Es posible eliminar la señal de **MemaReg** del control del uniciclo? En caso afirmativo, muestre o explique detalladamente las modificaciones necesarias. ¿Es posible eliminar alguna otra señal del control?
   c) ¿Es posible implementar en el uniciclo una instrucción **swap** que intercambie dos registros, haciendo sólo pequeñas modificaciones (como agregar señales de control, multiplexores o compuertas)?

d) Si por un error en la síntesis de la función del control del uniciclo, la señal **MemaReg** permanece siempre en 0, ¿qué instrucciones se verían afectadas y cuáles continúan operando correctamente?
e) Si ahora el error consiste en que la señal **MemaReg** tiene el valor invertido respecto a lo que especifica la tabla de verdad, ¿funcionarían correctamente algunas de las instrucciones? ¿Cuáles?
f) Si una estadística demuestra que es más utilizada la instrucción **bne** (saltar si distinto) que la **beq** (saltar si igual), ¿qué modificaciones realizaría al diseño del uniciclo sin aumentar la complejidad del mismo?

2) *[Recomendado]* Si los retardos de cada unidad funcional de la arquitectura uniciclo son los siguientes: 350 ps para la memoria de instrucciones, 400 ps para la memoria de datos, 180 ps para la ALU y los sumadores y 220 ps para el banco de registros, y utilizando gcc como programa de prueba, con la mezcla de instrucciones detallada en el siguiente cuadro:

Tipo de Instrucción	**% gcc**
Aritméticas	48 %
Transferencias de datos	33 %
Saltos condicionales	17 %
Saltos incondicionales	2 %

Se pide:
a) Si usted pudiese mejorar el retardo de un solo tipo de componente, reduciéndolo en un 10 %, ¿cuál componente elegiría? Justifique su respuesta.
b) Aplicando la mejora descrita en el apartado anterior, ¿cuánto sería la aceleración obtenida?
c) ¿Cuánto debería ser el retardo del componente seleccionado en el apartado a para que la aceleración total obtenida sea del 10 %?

3) Responda las siguientes preguntas, justificando en todos los casos su respuesta.
a) En el procesador segmentado visto en clase, no todas las instrucciones usan todas las etapas. Entonces, ¿es posible obtener algún beneficio haciendo que algunas instrucciones usen menos etapas?
b) ¿Es cierto que una ventaja de implementar segmentación en el camino de datos de un procesador es que esta es totalmente transparente para el programador?
c) ¿Es más conveniente buscar que el pipeline tenga mayor número de etapas de menor duración que intentar que las instrucciones demoren menos ciclos en ejecutarse?
d) ¿Con qué decisión de diseño se evitan los riesgos WAW en RISC-V?
e) ¿Por qué se afirma que aumentar el número de etapas del pipe incrementa el impacto de los riesgos?

4) *[Recomendado]* Sobre el mejor procesador visto en clase (segmentado con adelantamiento de datos) se ejecuta el programa gcc, que posee una mezcla de instrucciones mostrada en la siguiente tabla, y en el cual se supone que 2/3 de las instrucciones de transferencias de datos son cargas.

Tipo de Instrucción	**% gcc**	**/CPI**
Aritméticas	48 %	1,0
Transferencias de datos	33 %	1,4
Saltos condicionales	17 %	1,7
Saltos incondicionales	2 %	1,2

a) Sabiendo que el retardo de las memorias es de 200 ps, el retardo de la ALU y de los sumadores es de 100 ps, y que el del banco de registros es de 50 ps, calcule la productividad, factor de aceleración, rendimiento y latencia de una instrucción en este procesador. Tome como tiempo serie el del procesador uniciclo.
b) Repita el apartado anterior considerando en este caso que el retardo de la ALU es de 400 ps.

Laboratorio Ripes

5) Seleccione en el simulador Ripes la implementación de procesador de ciclo único (**Single Cycle Processor**). Dado el siguiente bloque de código:

```
addi x1, x0, 5
addi x2, x0, 10
add x3, x1, x2
lw x4, 0(x3)
```

   a) Ejecute el programa paso a paso e indique el valor del PC y de los registros involucrados en cada instrucción.
   b) Para cada instrucción, identifique qué unidades funcionales del camino de datos intervienen (memoria de instrucciones, banco de registros, ALU, memoria de datos, etc.).
   c) Indique cuál de las instrucciones determina el tiempo de ciclo del procesador y justifique su respuesta en función del camino crítico.
   d) Compare conceptualmente este comportamiento con el de un procesador segmentado. ¿Qué limitaciones presenta el procesador uniciclo?

6) Seleccione ahora la implementación (**5-stage processor w/o forwarding or hazard detection**). Dado el siguiente bloque de código:

```
add x1, x2, x3
add x4, x5, x6
add x7, x8, x9
```

   a) Ejecute el programa y complete un diagrama temporal indicando en qué etapa del pipeline (IF, ID, EX, MEM, WB) se encuentra cada instrucción en cada ciclo.
   b) Indique: -Cantidad total de ciclos necesarios para completar la ejecución. -CPI observado.
   c) Compare los resultados con el procesador uniciclo del ejercicio anterior.
   d) Explique por qué en este caso no aparecen riesgos.

7) Manteniendo la misma implementación (**5-stage processor w/o forwarding or hazard detection**) del ejercicio anterior. Ahora considere el siguiente bloque de código:

```
add x2, x5, x4
add x4, x2, x5
sw x5, 100(x2)
add x3, x2, x4
lw x3, 100(x4)
```

   a) Identifique todas las dependencias de datos presentes en el código.
   b) Indique cuáles de ellas generan riesgos de tipo RAW.
   c) Ejecute el programa y observe el resultado obtenido. ¿El resultado es correcto? Justifique
   d) Modifique el código insertando instrucciones nop de manera de obtener resultados correctos.
   e) Indique: -Cuántos nop fueron necesarios. -En qué posiciones se insertaron. -Qué dependencia obliga a cada inserción.

8) Seleccione ahora la implementación(**5-stage processor w/o hazard detection**). Dado el siguiente bloque de código:

```
add x1, x2, x2
add x3, x2, x1
add x4, x1, x3
```

   a) Ejecute el programa y observe el comportamiento del pipeline.
   b) Indique: -Qué dependencias se resuelven mediante adelantamiento. -Entre qué etapas se realiza el forwarding en cada caso.
   c) ¿Se producen errores en la ejecución? Justifique.
   d) Compare la cantidad de ciclos con el caso sin forwarding.

9) Seleccione ahora la implementación (**5-Stage processor w/o forwarding unit**). Dado el siguiente bloque de código:

```
add x1, x2, x3
sub x4, x1, x5
```

a) Ejecute el programa y observe el comportamiento del pipeline.
b) Indique: -Si se insertan stalls automáticamente. -Cuántos ciclos adicionales aparecen.
c) Explique el funcionamiento de la unidad de detección de riesgos en este caso.
d) Compare con la solución manual mediante nop del ejercicio 8.

10) Seleccione ahora la implementación (**5-stage processor**). Dado el siguiente bloque de código:

```
lw x1, 0(x2)
add x3, x1, x4
```

a) Ejecute el programa y observe el pipeline.
b) Indique: -Si se producen stalls. -Cuántos ciclos adicionales son necesarios.
c) Explique por qué este caso no puede resolverse completamente mediante adelantamiento.
d) Compare este comportamiento con el caso del ejercicio 9.

Ejercicio a Desarrollar en Clase Práctica

11) La empresa en la que usted trabaja decide diseñar un procesador (el AC19) parecido al RISC-V pero con un ISA que considera superior, el cual posee únicamente las siguientes diferencias:

- Instrucciones de procesamiento que acceden a memoria (Tipo Reg-Mem). Se habilitan todas las instrucciones de procesamiento para que uno de los operandos fuente pueda estar en memoria. La empresa considera que esto va a ser la ventaja competitiva frente al RISC-V.
- Para mantener la simplicidad y la misma cantidad de unidades funcionales que el RISC-V, se define que el único modo de direccionamiento a memoria será de registro indirecto, eliminando el modo de direccionamiento indexado. Este modo de direccionamiento se utilizará en las instrucciones load, store, y en las nuevas de tipo Reg-Mem.

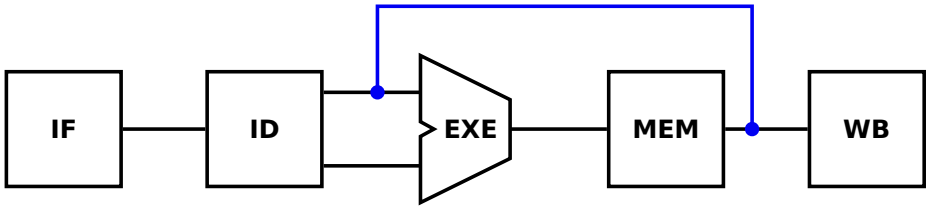
En este ejercicio se debe diseñar un procesador que ejecute el siguiente subconjunto del ISA del AC19:

Instrucción	**Comentario**
`ld r1, (r2)`	R[r1] ← M[R[r2]]
`st r1, (r2)`	M[R[r2]] ← R[r1]
`add r1, r2, (r3)`	R[r1] ← R[r2] + M[R[r3]]
`add r1, r2, r3`	Idéntica al RISC-V
`ori r1, r2, cte`	Idéntica al RISC-V
`bne r1, r2, cte`	Idéntica al RISC-V

a) Dibuje en detalle el camino de datos de un procesador uniciclo diseñado para implementar el ISA propuesto.
b) Dibuje las etapas que implementan el pipeline. Sólo se pide que haga un diagrama esquemático (como cajas negras), y no en forma detallada. Realice el diseño adecuado para que no haya riesgos estructurales, y demuéstrelo. Analice los riesgos de control y proponga una manera de minimizarlos.
c) Indique si en las siguientes secuencias de dos instrucciones habrá un riesgo RAW y esquematice los caminos de adelantamiento que se deben agregar para solucionarlos. Para cada caso, considere también la posibilidad de que haya instrucciones intermedias entre las dos planteadas.

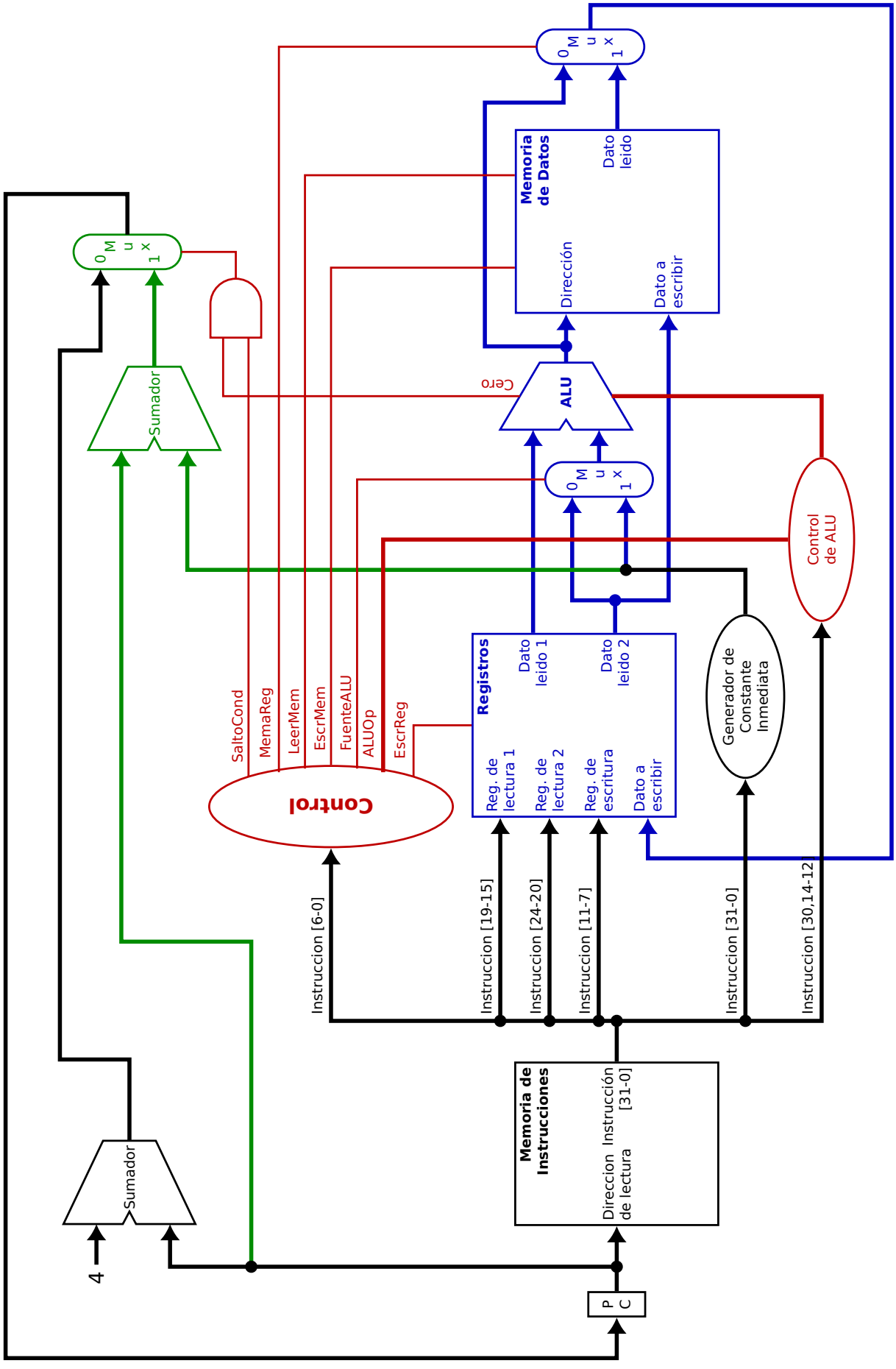
Secuencia	**Instrucciones**	**Secuencia**	**Instrucciones**
Caso 1	`add r1, r2, r3` `add r2, r1, r3`	Caso 2	`st r3, (r5)` `add r2, r1, (r5)`
Caso 3	`st r3, (r5)` `ld r4, (r5)`	Caso 4	`ld r3, (r5)` `add r1, r2, r3`
Caso 5	`ld r4, (r5)3` `st r4, (r5)`	Caso 6	`add r2, r1, (r5)` `add r3, r2, r1`

Los caminos de adelantamiento deben esquematizarse de una forma simple. Por ejemplo, en el RISC-V, un adelantamiento de la salida de memoria a la entrada de la ALU podría dibujarse de la siguiente manera:

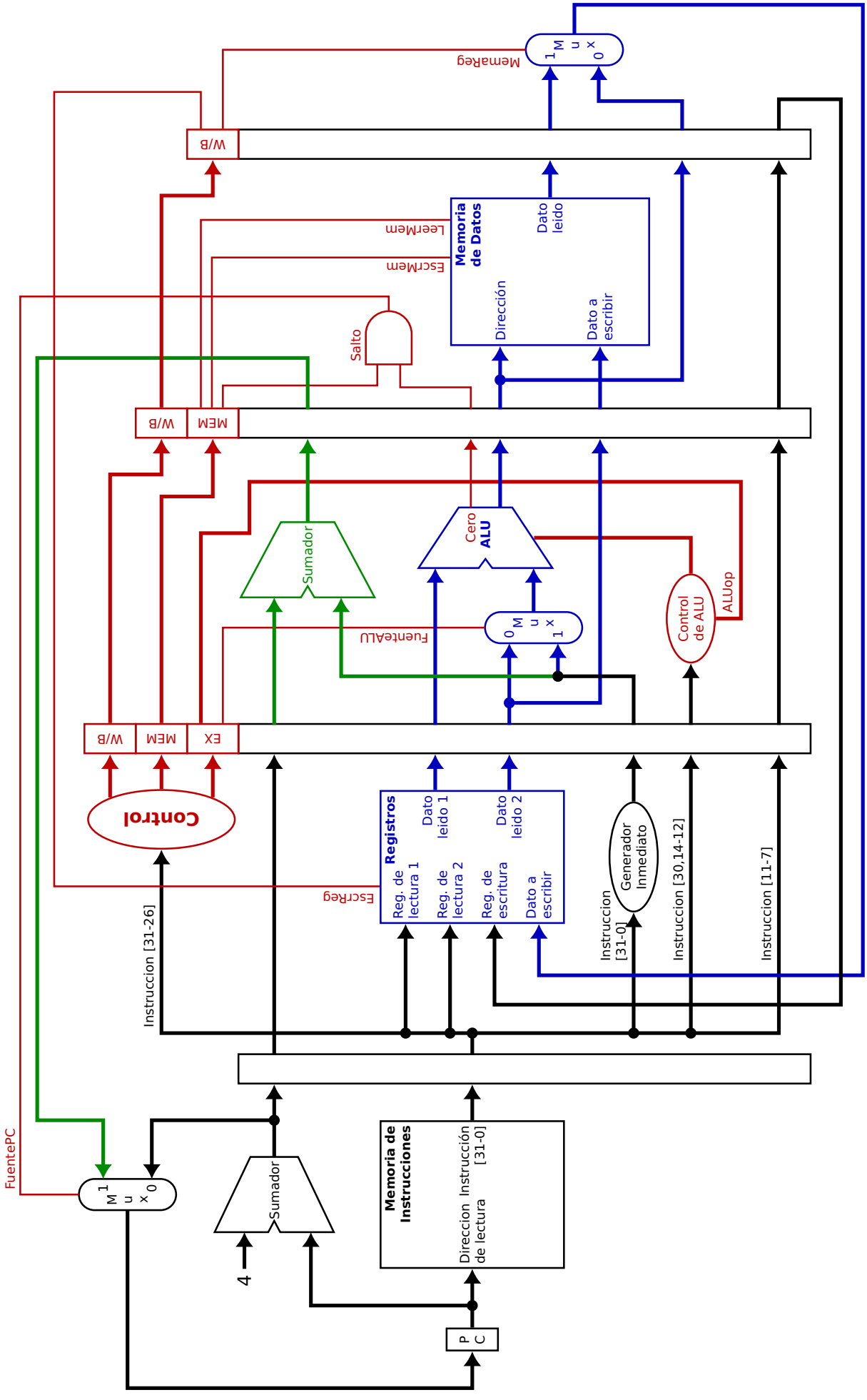


d) ¿Existirán riesgos WAW? Fundamente su respuesta. En caso afirmativo, escriba una secuencia de instrucciones para cada caso que contenga este riesgo (sin preocuparse por el sentido lógico que pueda tener la secuencia).
e) ¿Existirán riesgos WAR? Fundamente su respuesta. En caso afirmativo, escriba una secuencia de instrucciones para cada caso que contenga este riesgo (sin preocuparse por el sentido lógico que pueda tener la secuencia).

Esquema Procesador Uniciclo



Esquema Procesador Segmentado



Soluciones propuestas

1)
   a) ...
   b) **MemaReg** es idéntica que **LeerMem**. Además **ALUOP0** podría ser reemplazada por **SaltoCond**.
   c) No es posible de forma simple.
   d) Sólo **lw** se ve afectada.
   e) Las instrucciones **sw** y **beq** continúan operando correctamente.
   f) Colocar un negador en la señal **Cero** e implementar **bne** en lugar de **beq**.

2)
   a) Banco de Registros.
   b) 1,033.
   c) 158 ps.

3)
   a) No.
   b) No.
   c) Si.
   d) ...
   e) ...

4) a) Productividad = 1 / 1,255T; Aceleración = 2,39; Rendimiento = 79,68 %; Latencia = 1 ns
   b) Productividad = 1 / 1,255T; Aceleración = 1,79; Rendimiento = 79,68 %; Latencia = 2 ns

5)
   a) PC: 0,4,8,12
      x1=5, x2=10, x3=15, x4=Mem[15]
   b) IF, ID, EX, MEM, WB (según instrucción; lw usa todas)
   c) lw
   d) Mayor latencia, CPI=1 fijo, no hay solapamiento

6)
   a) I1: 1-5, I2: 2-6, I3: 3-7
   b) Ciclos=7, CPI=7/3≈ 2.33
   b) Mayor cantidad de ciclos que pipeline ideal
   c) No hay dependencias

7)
   a) (x2,x4),(x2),(x4),(x3)
   b) Todas son RAW
   c) Incorrecto
   d) Código con nops
   e) 6 nop
      add→add:2, add→sw:2, add→add:1, add→lw:1

8)
   a) Ejecución con forwarding
   b) EX→EX, MEM→EX
   c) Correcto
   d) Ciclos=7

9)
   a) Ejecución con stalls automáticos
   b) Sí, 2 ciclos adicionales
   c) Stall por dependencia RAW
   d) Igual que 2 nop manuales

10)
   a) Ejecución con forwarding + hazard detection
   b) Sí, 1 stall
   c) Load-use hazard
   d) Mejor que ejercicio 9