Universidad Católica San Pablo | Departamento de Ciencia de la Computación

Arquitectura de Computadores 2020-01 (CCOMP3-1.x)
Laboratorio 06b: Arquitectura en Multiciclo

Yván Jesús Túpac Valdivia
Universidad Católica San Pablo, Arequipa – Perú

10 de junio de 2020

---

1 Objetivos

- Practicar con la arquitectura Multiciclo para la ejecución de instrucciones MicroMIPS

2 Contenido base

- Diapositivas Ruta de datos y control: Multiciclo
- Videos de clase (incluido el de Microinstrucciones/Excepciones)
- Libro guía [Parhami, 2005]

Todos los alumnos antes de la sesión de laboratorio deberán revisar las clases de Ruta de datos y control en el material de clases y en [Parhami, 2005]

3 Actividades

3.1 Ejecución de intrucciones en multiciclo

Basado en la descripción de pasos de la Figura 1 y la Tabla 1 de la arquitectura multiciclo y las condiciones de la señales en cada ciclo de reloj:

- Indique la configuración de las señales de control en cada ciclo de reloj para la ejecución de las siguientes instrucciones:
  - `jal proc`
  - `beq rs, rt, dest`
  - `jr rs`
- Escriba las secuencias de microinstrucciones para estas tres instrucciones indicadas

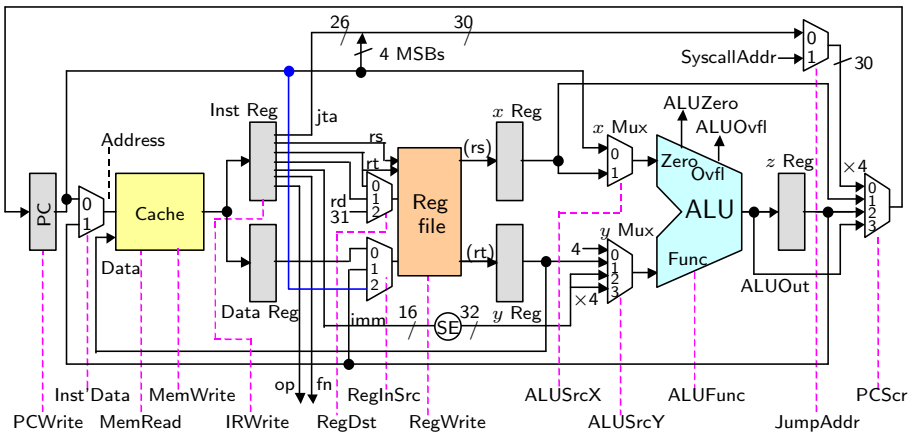

Figure 1: Arquitectura de ejecución en Multiciclo

3.2 Modificaciones a la ruta de datos en multiciclo

Qué cambios debería hacerse en la Tabla 1anterior (si fueran necesarios) si deseamos que la ruta de datos en multiciclo pueda ejecutar de forma nativa las siguientes pseudoinstrucciones MIPS:

- `move rs, rt`
- `blt rs, rt, dest`
- `sge rd, rs, rt`

3.3 Rendimiento de la arquitectura Multiciclo

Calcule el CPI (*Ciclos de reloj por instrucción*) que se alcanzaría si la arquitectura en Multiciclo pasa a utilizar la lógica de "próxima dirección" lo cual permitiría resolver saltos incondicionales y bifurcaciones ya en el 2do ciclo de reloj. Note que en la arquitectura multiciclo estudiada bifurcaciones y saltos se resuelven en el 3er ciclo de reloj. Considere los siguientes porcentajes de instrucciones para el cálculo:

Tipo ALU	44%	(4 ciclos de reloj)
Load	24%	(5 ciclos de reloj)
Store	12%	(4 ciclos de reloj)
Branch	18%	(3 $\rightarrow$ 2 ciclos de reloj)
Jump	02%	(3 $\rightarrow$ 2 ciclos de reloj)

3.4 Rendimiento de la arquitectura multiciclo - caso real

Con los CPI de la Tabla anterior (tanto para 3 y 2 ciclos en saltos y bifurcaciones), calcule:

- El total de ciclos de reloj utilizados
- El CPI promedio
- La diferencia porcentual de CPI promedio (si se usa 2 ciclos o 3 ciclos para los saltos y bifurcaciones)

		Instruction	Operations	Signal settings
Fetch & PC incr	1	Todas	Read out the instruction and write it into instruction register, increment PC	Inst'Data=0, MemRead=1 IRWrite=1, ALUSrcX=0 ALUSrcY=0, ALUFunc='+' PCSrc=3, PCWrite=1
Decode & reg read	2	Todas	Read out `rs` & `rt` into x & y registers compute branch address and save in z register	ALUSrcX=0, ALUSrcY=3 ALUFunc='+'
ALU oper & PC update	3	ALU type	Perform ALU operation and save the result in z register	ALUSrcX=1,ALUSrcY=1/2 ALUFunc: from instruction
		Load/Store	Add base and offset values, save in z register	ALUSrcX=1, ALUSrcY=2 ALUFunc='+'
		Branch	If (x reg)=≠<(y reg), set PC to branch target address	ALUSrcX=1, ALUSrcY=1 ALUFunc='-', PCSrc=2 PCWrite=ALUZero or ALUZero' or $ALUOut_{31}$
		Jump	Set PC to the target address `jta`, `SysCallAddr` or `(rs)`	JumpAddr=0/1 PCSrc=0/1, PCWrite=1
Reg write or mem access	4	ALU type	Write back z reg into `rd`	RegDst=1, RegInSrc=1 RegWrite=1
		Load	Read memory into data reg	Inst'Data=1, MemRead=1
		Store	Copy y reg into memory	Inst'Data=1, MemWrite=1
Reg write for `lw`	5	Load	Copy data register into `rt`	RegDst=0, RegInSrc=0 RegWrite=1

Table 1: Configuración de las señales de control

para la ejecución del siguiente código, desde que inicia la primera instrucción hasta el final.

Considere que las psudoinstrucciones son convertidas en una o más instrucciones reales (hacer el calculo con las instrucciones reales) y, de haber bucles, deben considerarse todas sus repeticiones (desdoblar el código).

```
# Rutina para hallar el mayor entero de la lista values y
# colocarlo en $t0.
      la   $s1,values      # remember how change this pseudo-instruction
                           # in real MIPS instructions
      lw   $t0,0($s1)      # initialize maximum to A[0]
      la   $s1,size
      lw   $s2,0($s1)
      addi $t1,$zero,0     # initialize index i to 0
loop: add  $t1,$t1,1       # increment index i by 1
      beq  $t1,$s2,done    # if all elements examined, quit
      add  $t2,$t1,$t1     # compute 2i in $t2
      add  $t2,$t2,$t2     # compute 4i in $t2
      add  $t2,$t2,$s1     # form address of A[i] in $t2
      lw   $t3,0($t2)      # load value of A[i] into $t3
      slt  $t4,$t0,$t3     # maximum < A[i]?
      beq  $t4,$zero,loop  # if not, repeat with no change
      addi $t0,$t3,0       # if so, A[i] is the new maximum
      j loop               # change completed; now repeat
done: ...                        # continuation of the program

.data
values: .word 3,10,15,9,7,12,9,20
size:   .word 6
```

References

[Parhami, 2005] Parhami, B. (2005). *Computer Architecture: From Microprocessors to Supercomputers*. The Oxford Series in Electrical and Computer Engineering. OUP USA.