

PRÁCTICA 2

Por Pablo Amor Molina 100495855 y
Manuel Roldán Matea 100500450

Ejercicio 1

	Diseño	Señales de control	Comentarios
lc R1 R2 (R3)	MAR <- PC MBR <- memory[MAR] PC <- PC + 4 R1 <- MBR Salto a fetch	(C0, T2) (R, BW=11, M1, C1, TA, M2, C2) (SelC=10101, LC, T1, A0=1, B=1, C=0)	Hay que tener en cuenta la suma al PC ya que la parte imaginaria se encuentra en la dirección de memoria del contador de programa por lo que hay que sumar 4 a PC para apuntar a la siguiente instrucción
sc R1 R2 (R3)	MAR <- R3 MBR <- R1 memory[MAR] <- MBR MAR <- R3 + 4 MBR <- R2 memory[MAR] <- MBR Salto a fetch	(MR=0, SelA=1011, T9, C0) (MR=0, SelA=10101, T9, C1) (TA, TD, W, BW=11) (MB=10, MC=1, selCOP=1010, T6, C0, SelA=1011) (MR=0, SelB=10000, T10, C1) (TA, TD, W, BW=11, A0=1, B=1, C=0)	- Nada a destacar
lc R1 R2 (R3)	MAR <- R3 MBR <- memory[MAR] R1 <- MBR MAR <- R3 + 4 MBR <- memory[MAR] R1 <- MBR Salto a fetch	(MR=0, SelA=1011, T9, C0) (TA, R, BW=11, M1, C1) (T1, LC, SelC= 10101) (MB=10, MC=1, selCOP=1010, T6, C0, SelA=1011) (TA, R, BW=11, M1, C1) (T1, LC, SelC= 10000, A0=1, B=1, C=0)	- Nada a destacar
addc R1 R2 R3 R4	R1 <- R1 + R3 Actualizo SR R2 <- R2 + R4 Salto a fetch	(SelA=10101, SelB=1011, MC, SelCop=1010, T6, SelC=10101, LC, SelP=11, M7, C7) (SelA=10000, SelB=110, MC, SelCop=1010, T6, SelC=10000, LC, A0=1, B=1, C=0)	- Se debe actualizar sr en el mismo ciclo que la primera suma.
mulc R1 R2 R3 R4	RT1 <- R1 * R3 RT2 <- R2 * R4 RT3 <- RT1 - RT2 actualizo sr RT1 <- R1 * R4 RT2 <- R2 * R3 R2 <- RT1 - RT2 R1 <- RT3 Salto a fetch	(selA=10101, selB=1011, selCop=01100, MC, T6, C4) (selA=10000, selB=110, selCop=01100, MC, T6, C5) (selCop=01011, MC, MA, MB=01, C6 selP=11, M7, C7) (selA=10101, selB=110, selCop=01100, MC, T6, C4) (selA=10000, selB=1011, selCop=01100, MC, T6, C5) (selC=10000, LC, selCop=01011, MC, T6, MA, MB=01) (T7, SelC=10101, LC, A0=1, B=1, C=0)	- Es necesario guardar R1 en un RT para poder realizar todas las operaciones y no operar con R1 cambiado por las primeras operaciones.

beqc R1 R2	MBR <- SR SR (R1 - R3) IF SR.Z != 0, j beq2 SR (R2 - R4) IF SR.Z != 0, j beq2 RT1 <- PC RT2 <- S6 PC <- PC + S6 beq2: SR <- MBR Salto a fetch	(T8,C1) (SelA=10101, SelB=1011, SelCop=01011, MC, SelP=11, M7, C7) (A0=0, B=1, C=110, MADDR=beq2) (SelA=10000, SelB=110, SelCop=01011, MC, SelP=11, M7, C7) (A0=0, B=1, C=110, MADDR=beq2) (T2, C4) (SE=1, Offset=0, Size=110, T3, C5) (MA, MB=01, SelCop=1010, MC, T6, C2) beq2: (T1, C7, A0=1, B=1, C=0)	- Hay que destacar que al ser 2 condiciones se deben realizar 2 saltos y 2 comprobaciones, además de guardar el sr en mbr antes de todo para que al final se pueda recuperar.
call U20	ra(x1) <- PC PC <- U20 Salto a fetch	(LC, SelC=1, MR, T2) (SIZE=10100, OFFSET=0, SE=0, T3, C2, A0=1, B=1, C=0)	- Se debe utilizar MR puesto que el ID del registro no viene dado por la instrucción IR.
ret	PC <- ra(x1) Salto a fetch	(SelA=0, MR, T9, C2, A0=1, B=1, C=0)	- Se puede lograr en un solo ciclo
hcf	PC <- x0 SR <- x0 Salto a fetch	(SelA=0, MR, T9, C2) (SelA=0, MR, T9, C7, A0=1, B=1, C=0)	- Hay que seleccionar x0 para asegurar que lo que llevamos a ambos registros es un0

Ejercicio 2

	Ciclos de reloj Sin extensión	Ciclos de reloj con extensión	Mejora (%)
A==B	131 ciclos	102 ciclos	19,84 % mejora
A!=B	105 ciclos	92 ciclos	12,38% mejora

La mejora de ciclos respecto al código sin extensión se debe a que este no contiene instrucciones para manejar dos registros a la vez, es decir, la parte real y la imaginaria trabajan de manera independiente, teniendo que ejecutar primero la real y justo después la imaginaria por lo que el programa trabaja el doble.

Sin embargo, la parte con extensión se crean instrucciones específicas para complejos, es decir, para operar la parte real y la imaginaria de manera simultánea y de ese modo el procesador necesita menos ciclos.

Después de hacer la comparación se puede ver que con la extensión el programa es mucho más eficiente por lo tanto si necesitamos realizar cualquier instrucción que conlleve usar complejos es rentable usar la extensión.

Conclusiones y problemas encontrados

Gracias a esta práctica hemos podido conocer una pequeña introducción a la microprogramación, lo que nos ha sido de gran ayuda para esta parte del curso. Gracias a esta misma hemos solventado con bastante soltura el ejercicio relativo al tema del último mini examen, por lo que estamos bastante satisfechos.

Los principales problemas que nos encontramos estuvieron principalmente localizados en el inicio de la práctica, pues no habíamos trabajado nunca con la herramienta y estábamos bastante desorientados. Comenzamos haciendo las instrucciones sc y lc, pero nos costó mucho más sacar la instrucción lc, puesto que no sabíamos trabajar con las dos palabras hasta que el coordinador, nuestro profesor en las clases magistrales, nos pudo dar unas breves indicaciones. A partir de ahí el desarrollo del trabajo ha sido bastante llevadero y progresivo.

	Horas estimadas
Ejercicio 1	11h
Ejercicio 2	3h
Memoria	2:30h