MEMORIA PRÁCTICA 2
ESTRUCTURA DE COMPUTADORES

-INTRODUCCIÓN A LA MICROPROGRAMACIÓN-

Autores:

Ismael Molinero Sánchez	NIA: 100405964	GRUPO 80
Javier Sánchez-Majano Sánchez	NIA: 100406003	GRUPO 80

Ingeniería informática Colmenarejo

ÍNDICE

EJERCICIO 1:

Nombre de la instrucción	Diseño de instrucciones en lenguaje RT	Señales de control a activar	Decisiones de diseño
ld RDEST, RORIG	RDEST←RORIG	(SELA=10000, T9, SELC=10101, LC=1, A0=1, B=1, C=0)	Es un diseño simple en el que la que Rorig era cargado y desplazado a Rdest
ldi RDEST, U16	RDEST←SELEC	(SE=0, OFFSET=0, SIZE=10000, T3=1, LC=1, MR=0, SELC=10101, A0=1, B=1, C0=1)	Es un diseño simple en el que se carga un valor inmediato y es destinado a Rdest
ld RDEST, (RORIG)	MAR←RORIG Memory←MAR MBR←Memory RDEST←MBR	(MR=0, SELA=10000, T9=1, C0=1) (TA=1, R=1, BW=11, SE=1, M1=1, C1=1) (T1=1, LC=1, MR=0, SELC=10101, SE=1, A0=1, B=1, C=0)	En el primer ciclo se carga en MAR el Rorig, y pasa, en el segundo ciclo su dirección a memoria, y carga el valor del dato asociado a esa dirección en MBR. En el último ciclo, lo carga en Rdest
add_a RORIG	RT1←RORIG RT2←A A←RT1+RT2	MR=0, SELA = 10101, T9=1, C4=1) (MR=1, SELB = 100, T10=1, C5=1) (MC=1, MR=1, MA=1, MB=1, SELCOP=1010, T6=1, SELC=100, LC=1, SELP=11, M7, C7, A0=1, B=1, C=0)	En el primer ciclo se carga en RT1 Rorig, y en el segundo se carga A en RT2. Para finalizar, en el último ciclo, suma RT1 y RT2 y guarda el valor en A

addi_a S16	RT2←SELEC RT1←A A←RT1+RT2	(SE=1, OFFSET=0, SIZE=10000, T3=1, C5=1) (MR=1, SELA=100, T9=1, C4=1) (MC=1, MR=1, MA=1, MB=1, SELCOP=1010, T6=1, SELC=100, LC=1, SELP=11, M7, C7, A0=1, B=1, C=0)	En el primer ciclo, carga SelC en RT2. En el segundo ciclo carga en RT1 A. Y para finalizar, en el último ciclo, suma RT1 y RT2 y lo guarda en A
inc RDEST	RDEST ←RDEST+1	(MC=1, MR=0, SELA=10101, MA=0, MB=11, SELCOP=1010, T6=1, SELC=10101, LC=1, SELP=11, M7, C7, A0=1, B=1, C=0)	Es un código simple en el que a Rdest se le suma 1 y lo devuelve a Rdest
dec RDEST	RDEST←RDEST-1	(MC=1, MR=0, SELA=10101, MA=0, MB=11, SELCOP=1011, T6=1, SELC=10101, LC=1, SELP=11, M7, C7, A0=1, B=1, C=0)	Es un código simple en el que a Rdest se le resta 1 y lo devuelve a Rdest
jp S16	RT1←PC RT2←SELEC PC←RT1+RT2	(T2=1, C4=1) (SE=1, OFFSET=0, SIZE=10000, T3=1, C5=1) (MA=1, MB=01, MC=1, SELCOP=1010, T6=1, M2=0, C2=1, A0=1, B=1, C=0)	En el primer ciclo, carga PC en RT1. En el segundo ciclo, carga la dirección de memoria en RT2. Para finalizar, en el último ciclo, suma RT1 y RT2 y lo devuelve a PC

jpz S16	if (Z == 0) {
	RT1←PC
	RT2←SELEC
	PC←RT1+RT2
	}
		(A0=0,B=1,C=110,MADDR=FTCH)
		(T2=1, C4=1)
		(SE=1, OFFSET=0, SIZE=10000, T3=1, C5=1)
		(MA=1, MB=01, MC=1, SELCOP=1010, T6=1, M2=0, C2=1)
		FTCH: (A0=1, B=1, C=0)
			En el primer ciclo, se comprueba que Z sea igual a 0.
			Si se cumple, en el segundo ciclo, carga el valor de PC en RT1.
			En el tercer ciclo carga la dirección relativa en RT2.
			Y en el último ciclo, suma RT1 con RT2 y devuelve el resultado a PC

call U16	SP ← SP - 4 MBR←PC MAR←SP Dir.Memoria←MAR Memoria(SP)←MBR PC←SELEC	(MC=1, SELA=11101 MR=1, MA=0, MB=10, SELCOP=1011, T6=1, LC=1, SELC=11101) (T2=1, M1=0, C1=1) (MR=1, SELB=11101, T10=1, C0=1) (TA=1, TD=1, W=1, BW=11) (SE=0, OFFSET=0, SIZE=10000, T3=1, M2=0,C2=1, A0=1, B=1, C=0)	En el primer ciclo saca el valor de la pila. En el segundo ciclo pasa el valor de PC a MBR. En el tercer ciclo pasa el valor de la pila a MAR. En el cuarto ciclo, pasa MAR a la dirección de memoria. En el quinto ciclo de MBR pasa a la memoria. Y para finalizar Selec pasa a PC.
ret	MAR←SP RT1←SP Memoria←MAR MBR←Memoria PC←MBR PC0←U16	(MR=1,SELA=11101,T9=1, C0=1, C4=1), (TA=1, R=1, BW=11, SE=1, M1=1, C1=1), (T1=1, M2=0, C2=1), (MC=1, MR=1, MA=1, MB=10, SELCOP=1010, T6=1, LC=1, SELC=11101, A0=1, B=1, C=0)	En el primer ciclo pasa de la pila a MAR. En el segundo ciclo pasa de SP a RT1. En el tercer ciclo pasa de MAR a memoria. En el cuarto ciclo pasa de la memoria a MBR. En el quinto ciclo pasa de MBR a PC. Para finalizar guarda el valor en PC0.

halt	PC←EXCODE=0	(EXCODE=0, T11=1, M2=0, C2=1)	En esta función,se pone el PC a 0 a través de la señal de control ExCode
push RORIG	SP ← SP – 4 MBR←RORIG MAR←SP Dir.Memoria←MAR Memoria(SP)←MBR	(MC=1, SELA=11101 MR=1, LC=1, MA=0, MB=10, SELCOP=1011, T6=1, SELC=11101), (MR=0, SELA=10101, T9=1, M1=0, C1=1), (MR=1, SELB=11101, T10=1, C0=1), (TA=1, TD=1, W=1, BW=11, A0=1, B=1, C=0)	En el primer ciclo, se hace un hueco en la pila En el segundo ciclo, MBR carga la dirección de Rorig. . En el tercer ciclo se carga en MAR la dirección. En el cuarto ciclo se pasa la dirección desde el MAR. Y en el último ciclo se guarda en la pila MBR.
pop RDEST	RT1←SP MAR←SP Memoria←MAR MBR←Memoria RDEST←MBR SP←SP+4	(MR=1,SELA=11101,T9=1, C0=1, C4=1), (TA=1, R=1, BW=11, SE=1, M1=1, C1=1), (T1=1, LC=1, MR=0, SELC=10101), (MC=1, MR=1, MA=1, MB=10, SELCOP=1010, T6=1, LC=1, SELC=11101, A0=1, B=1, C=0)	En el primer ciclo se busca la dirección en la pila. En el segundo ciclo MBR Lee la dirección de la memoria. En el tercer ciclo carga la dirección en Rdest. Y para finalizar le resta a la pila 4 para volverá a su origen.

EJERCICIO 2:

Las ventajas y desventajas de estos lenguajes son:

MIPS 32

VENTAJAS	DESVENTAJAS
Fácil comprensión	No están las instrucciones de pop y push
Más optimizado	Variaciones de rendimiento
Más funciones	

Z80

VENTAJAS	DESVENTAJAS
Están las instrucciones pop y push	Menos optimizado
Instrucciones más básicas y sencillas	Menos visual
	Funciones compuestas

Las diferencias que se pueden apreciar entre la ejecución del programa entre el conjunto de instrucciones Z80 y MIPS32 son:

- Tiempo de ejecución

	Z80	MIPS32
Instructions	111	74
CLK ticks	669	147

En MIPS32, se ejecuta más eficientemente, ya que para el mismo resultado, se necesitan 37 instrucciones más y 522 ciclos de reloj más que el programa con las instrucciones de Z80.

- Registros:
  En los dos lenguajes se utilizan los mismos registros para representar los datos, $a0=IX, $a1=IY y $v0=HL, pero sin embargo, el lenguaje Z80 necesita un registro extra, el registro A, el contador, algo que ya está implementado en MIPS32, lo que hace que sea más eficiente este lenguaje.

- Memoria
  Se utilizan distintas direcciones de memoria según lenguaje.
  También ocupa menos memoria con el lenguaje Z80 en main, sumav y f1, pero en b1, ocupa menos en MIPS32.
  En pila ocupa más en Z80.

En conclusión, en este programa, el lenguaje MIPS32 es mejor que el Z80.

CONCLUSIONES:

- Esta práctica es má visual que al anterior, por lo que se nos ha hecho más sencillo realizar las instrucciones, al saber cómo iban sucediendo en el procesador las microinstrucciones.
- La función de grabación ha hecho que sea más fácil comprobar que está bien el programa que se ha hecho si algo no funciona cuando lo corrige el profesor.
- Nos hubiera gustado que hubiera durado más la práctica ya que creemos que esta es una parte esencial dentro de la asignatura.