UNICEN
Universidad Nacional del Centro de la Provincia de Buenos Aires

Arquitectura de Computadoras I

Ingeniería de Sistemas – Curso 2025

Trabajo Práctico 1: Microprocesador MIPS

1. Siendo la siguiente instrucción escrita en alto nivel:

```
f = (g + h) - (i + j);
```

Las variables *f*, *g*, *h*, *i* y *j* se asignan a los registros *$s0*, *$s1*, *$s2*, *$s3* y *$s4*, respectivamente, describa la secuencia de instrucciones MIPS equivalente a la sentencia.

2. Considerando que las variables de sentencia condicional *f*, *g*, *h*, *i* y *j* se encuentran almacenadas en los registros comprendidos entre *$s0* y *$s4*, describa la secuencia de instrucciones MIPS equivalente a las siguientes sentencias:

```
if (i==j) f = g + h; else f = g-h
```

3. Siendo el siguiente bucle descrito en C:

```
while (save[i] == k) i += 1;
```

Suponiendo que *i* y *k* corresponden a los registros *$s3* y *$s5*, y la base del arreglo *save* está en *$s6*. ¿Cuál es el código ensamblador MIPS correspondiente a este segmento de código?

4. Se desea añadir al conjunto de instrucciones soportado por el procesador MIPS visto en el curso, la instrucción *JR*. Esta instrucción permite realizar un salto incondicional a la instrucción cuya dirección se encuentra almacenada en el registro *rs*, y es utilizada para el retorno en llamadas a rutinas. El formato de la instrucción es el siguiente:

31 – 26	25 – 21	20 – 16	15 – 11	10 – 6	5 – 0
SPECIAL 000000	**rs**	00000	00000	00000	**JR** 001001
6	5	5	5	5	6

Descripción: PC ← Reg[rs]

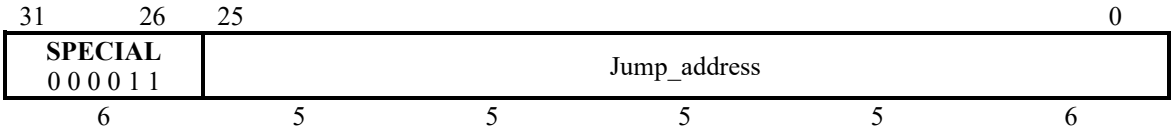
a) Realizar un diagrama esquemático del procesador que contenga las modificaciones necesarias para que el procesador pueda soportar dicha instrucción.
b) Determinar el valor para cada una de las señales de control (*RegWrite, AluSrc, AluOp, RegDst, MemWrite, MemRead, MemtoReg)*. Si es necesario incorporar nuevas señales de control, detallar también sus valores.

5. Considerando el siguiente código de una función:

```
int example (int g, int h, int i, int j)
{
      int f;
      f = (g+h) - (i+j);
      return f;
}
```

En donde los parámetros *g*, *h*, *i* y *j* se corresponden con los registros comprendidos entre *$a0* y *$a3*, que la variable de retorno *f* se encuentra asociada a *$s0*, y la dirección de retorno se encuentra almacenada en *$ra,* describa la secuencia necesarias de instrucciones MIPS para la ejecución de la función, considerando que el único registro que puede ser sobrescrito es el registro *$s0*. Nota: los registros auxiliares necesarios para el cálculo dentro de la subrutina deben ser almacenados y restaurados utilizando la pila de programa, apuntada por el registro *$sp*.

6. Se desea añadir al conjunto de instrucciones soportado por el procesador MIPS visto en el curso, la instrucción *JAL*. El formato de la instrucción es el siguiente:



Descripción: Reg[$ra] ← PC + 4
PC ← PC(31:28) & Jump_address & 00

La actualización de PC durante la ejecution de una instrucción *JALR* se realiza en la etapa de WB.

a) Realizar un diagrama esquemático del procesador que contenga las modificaciones necesarias para que el procesador pueda soportar dicha instrucción.
b) Determinar el valor para cada una de las señales de control (RegWrite, AluSrc, AluOp, RegDst, MemWrite, MemRead, MemtoReg). Si es necesario incorporar nuevas señales de control, detallar también sus valores.

7. Considerando las siguientes funciones:

```
int suma (int a, int b)
{
       Int c;
       c = a+b;
        return c;
}

int example (int g, int h, int i, int j)
{
       int f;
       f = suma(g,h) – suma(i,j);
       return f;
}
```

Describa el conjunto de instrucciones MIPS equivalente al código presentado, considerando que los parámetros de la función se encuentran comprendidos entre *$a0* y *$a3*, y que la variable *f* está asociada a *$s0*. Se debe considerar los llamados a funciones y además que ningún registro puede ser sobrescrito.

8. Considerando la siguiente función que realiza el cálculo del valor factorial de un número *n*, el cual se encuentra almacenado en el registro *$a0*

```
int factorial (int n)
{
	if (n < 1)
		return 1;
	else
		return (n*factorial(n-1));
}
```

Describa la secuencia de instrucciones MIPS equivalentes al código presentado.

9. Dada la siguiente instrucción de salto:

```
beq $s0, $s1, L1
```

a) Determine el rango de direcciones posibles de salto, asumiendo que la instrucción se encuentra almacenada en la dirección 0x0004C000
b) Reemplace la instrucción con un conjunto de instrucciones que permita extender el rango de salto.
c) Determine el rango de direcciones posibles de salto para el inciso b).