Patricio Damián Bruno DNI 42077426 LU 62/19

1)

a)

I) tamaño del PC: 32 bits, ya que el enunciado dice que la máquina direcciones de 32 bits y el PC almacena direcciones de memoria

tamaño máximo de la memoria: 2^32 direcciones * 1 (byte/direccion) = 2^32 bytes de memoria

cantidad de direcciones de memoria: cantidad de formas de combinar 32 bits = 2^32

II) El número máximo que podemos sumar a un registro mediante direccionamiento inmediato se puede calcular de la siguiente manera:

en el formato de instruccion 'l' hay 12 bits disponibles para almacenar el valor inmediato. Eso quiere decir que ahí entra un

número representado en complemento a 2 de 12 bits de largo. El número positivo más grande que podemos representar así es (2^12)-1 = 4095

b)

I) PC: es necesario (32 bits)

SP: no, ya que no hay CALL ni RET

[SP+1]: no es necesario IR: es necesario (32 bits)

Ode ottoren a delene de la instrucciona

2da y tercera palabra de la instruccion: no es necesario

flags: ninguno es necesario

registros: ahora hay 32 (R0 siempre vale 0)

II)

PC	SP		R0	R11	R12	R13							
0x00000014				0 0	1	0	70						
	+0	+4	+8	+C						ППП			
0x00000000	0x00158613	0x00164693	0x00D65463	0x00068733									
0x00000010	0x00D620A3	0xFE0688E3								Ш			
	PC	SP	[SP+1]	IR	i i	Instrucción primer palabra		PC	pala	Poal	Instrucciones decodificada		
1	0x00000000			0x00158613	00000000001 01011 000 01100 0010011			0x00000004			ADDI r12, r11, 0x001		
	Ejecución	r12 = r11 + 0x00000001											r12
2	0x00000004			0x00164693	00000	000 0001 0	1100 100 01101 0010011	0x00000008			XORI r13, r12,	0x001	
	Ejecución	r13 = r12 xor 0x00000001											r13
3	0x00000008			0x00D65463	0 00000	00 01101 01	100 101 0100 0 1100011	0x0000000C			BEG r12, r1	3, 8	
	Ejecución	si (r12 >= r13) : pc = pc + 8. Como se cumple la condición, se salta a 0x00000014											
4	0x00000014			0xFE0688E3	1 11111	1 00000 011	01 000 10000 1 1100011	0x00000018			BEQ r13, r0,	-16	
	Ejecución	si (r13 == r0) : pc = pc + (-16). Como se cumple la condición, se salta a 0x00000014											
5	0x00000008		1	0x00D65463	0 00000	00 01101 01	100 101 0100 0 1100011	0x0000000C			BEG r12, r1	3, 8	
	Ejecución	si (r12 >= r13) ; pc = pc + 8. Como se cumple la condición, se salta a 0x00000014						00014					
6	1												
	Ejecución	el programa va a ciclar infinitamente entre las posiciones de memoria 0x00000014 y 0x00000008											
7						•		i e		ПШ			

c) A = parte alta en r13, parte baja en r12 B = parte alta en r15, parte baja en r14

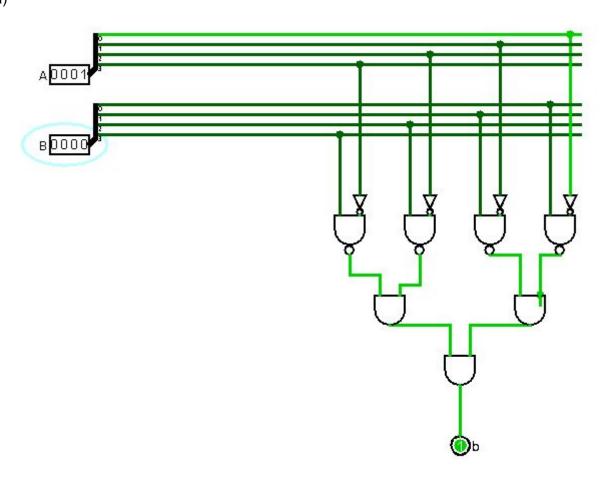
en r10 guardo la parte baja de la respuesta y en r11 la alta en r9 guardo el carry de la primera suma me doy cuenta si hay carry porque en caso de haberlo (asumiendo números sin signo),

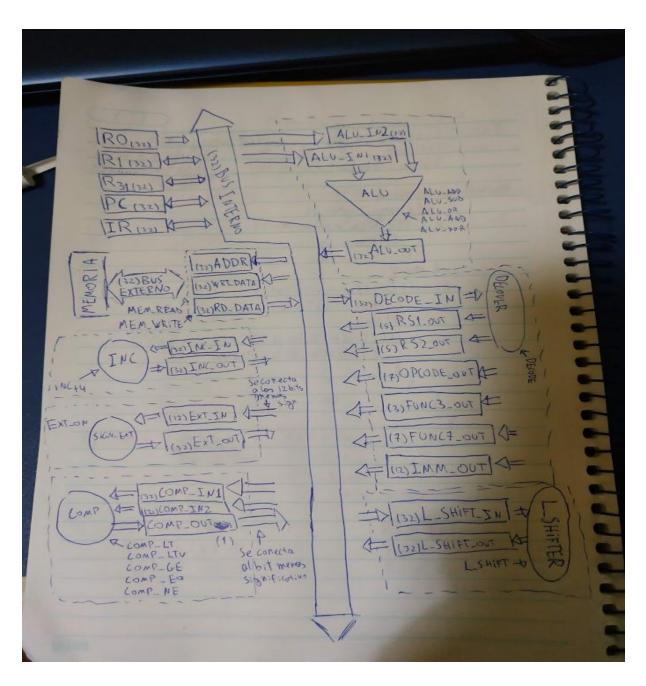
el resultado sería menor a alguno de sus sumandos (que no tiene sentido al sumar números positivos)

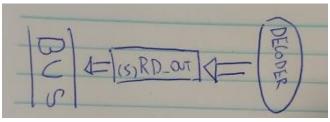
add r10, r12, r14 sltu r9, r10, r12 ;me fijo si r10 es menor a r12 add r11, r13, r15 add r11, r11, r9 ;sumo el carry

2)

a)







Aclaraciones:

- Todos los registros que guardan direcciones de memoria o palabrasson de 32 bits
- El shifter es para salvar la excepción del bit 0 del dato Imm en las instrucciones de formato B
- Todos los registros de salida del decoder de n bits están conectados a los n bits menos significativos del bus interno.

Esto también aplica para el registro EXT IN y COMP OUT

- En el datapath yo dispuse todos los registros R1...31 sueltos y conectados directamente al bus por simplicidad.
- En realidad, la idea es que uno le provee un índice de registro al control unit (ese dato saldría de los registros RS1_OUT, RS2_OUT y RD_OUT del decoder).
- Luego, para leer, con un multiplexor se decide qué dato dejar salir al bus y para escribir, con un decodificador se decide qué enable_wrt habilitar, como en el taller.
 c) XOR:

```
ALU IN1 = RS1
ALU IN2 = RS2
ALU OR
RD = ALU OUT
BGE:
COMP IN1 = RS1
COMP_IN2 = RS2
COMP GE
if(COMP_OUT){ //COMP_OUT es de un solo bit
ALU IN1 = PC
EXT IN = IMM OUT
EXT ON
L_SHIFT_IN = EXT_OUT //porque en imm estan los bits 1...12 y sabemos que el bit
L SHIFT
ALU IN2 = L SHIFT OUT
ALU ADD
```

ADDI: ALU_IN1 = RS1 EXT_IN = IMM_OUT

PC = ALU_OUT

0 es 0

```
EXT_ON
      ALU_IN2 = EXT_OUT
      ALU ADD
      RD = ALU_OUT
3)
a)
TEMP -> 0x00000800 (lectura)
REFR -> 0x00000804 (escritura)
1 grado -> salto de 4095/127 unidades
main(){
      R1 = 0x000003C7 //0°C
      R2 = 0x000008D2 //41°C
      while(true){
             if([TEMP] >= R2){
                    [REFR] = 0x0000000F
             } else if([TEMP] < R1){
                    [REFR] = 0x000000000
             }
      }
}
main:
      R1 = 0x000003C7
      R2 = 0x000008D2
      R3 = 0x00000800 ; TEMP
      R4 = 0x00000804; REFR
loop:
      LW R5, 0(R3) ;uso R5 para hacer operaciones simples
      BGE R5, R2, encender
      BLT R5, R1, apagar
      BEQ R0, R0, loop
encender:
      ADDI R5, R0, 0x00F
      SW R5, 0(R4)
      BEQ R0, R0, loop
apagar:
      ADDI R5, R0, 0x000
      SW R5, 0(R4)
      BEQ R0, R0, loop
b)
rai(){
      [REFR] = 0x000000000
}
main(){
      R1 = 0x000003C7 //0°C
```

```
R2 = 0x000008D2 //41°C
      while(true){
             if([TEMP] >= R2){
                    [REFR] = 0x0000000F
             }
      }
}
rai:
      ADDI R5, R0, 0x000
      SW R5, 0(R4)
main:
      R1 = 0x000003C7
      R2 = 0x000008D2
      R3 = 0x00000800; TEMP
      R4 = 0x00000804; REFR
loop:
      LW R5, 0(R3) ;uso R5 para hacer operaciones simples
      BGE R5, R2, encender
      BEQ R0, R0, loop
encender:
      ADDI R5, R0, 0x00F
      SW R5, 0(R4)
      BEQ R0, R0, loop
```