NOMBRE: CARLOS EDUARDO SANCHEZ TORRES

EJERCICIO 1

Convertir el siguiente código a:

a) Diagrama de Flujo
b) Ensamblador MIPS
c) Lenguaje máquina

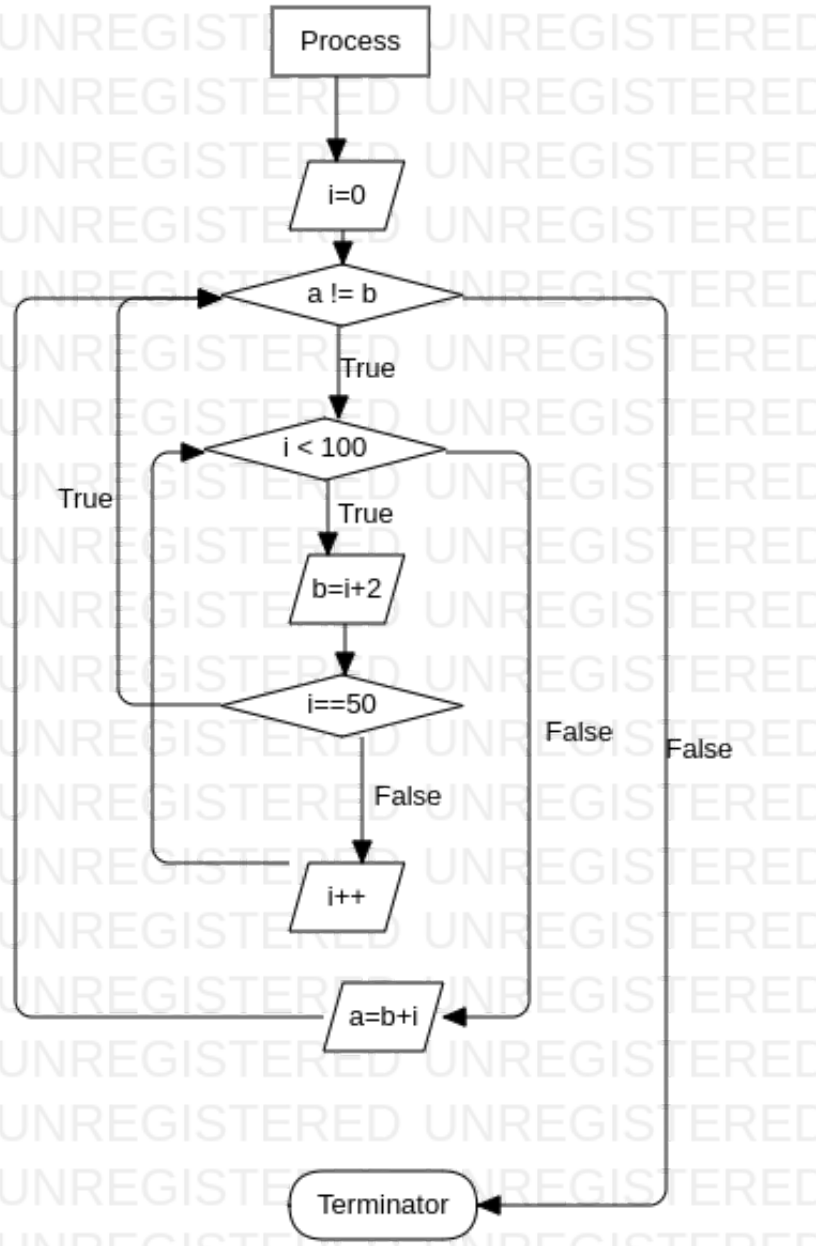
Suponga :

a=$t0, b=$S0, i= $t3, 100=$t5, 2=$t4, 1=$s5, 50=$s4,

y la dirección de inicio es $200_{10} = 11001000_2$

```
200:    i=0;
        while (a ≠ b)
         {
          while (i < 100)
           {
             b = i+2;
             if(i==50) break;
             i++;
           }
           a = b + i;
        }
```

a) Diagrama de Flujo



Valor 1, 70%; 2, 30%

b) Ensamblador MIPS

#CELDA	ETIQ	Código	Comentario
200		add $t3, $0, $0	# i=0
232	WHILE_1	beq $t0, $s0, EXIT_WHILE_1	# a == b => EXIT
264	WHILE_2	beq $t3, $t5, EXIT_WHILE_2	# i == 100 => EXIT
296		add $s0, $t3, $t4	# b = i+2
328		beq $t3, $s4, EXIT_WHILE_2	# i == 100 => EXIT_WHILE_2
360		add $t3, $t3, $s5	# i ++
392		j WHILE_2	
424	EXIT_WHILE_2	add $t0, $s0, $t3	# a = b + i
456		j WHILE_1	# Go to WHILE_1
488	EXIT_WHILE_1		

Valor 1, 70%; 2, 30%

Nota bene: !(i <100) = $i \geq 100$, pero como el incremento es 1 en 1, el bucle finaliza cuando i=100.

c) Lenguaje máquina

Ver **anexos** para el diccionario de registros a binario de anexos.

Ensamblador	Máquina
add RD RS RT	000000 RS RT RD 00000 100000
beq RS RT ADDRESS	000100 RS RT ADDRESS
j ADDRESS	000010 ADDRESS

Algoritmo

1. Diccionario de ensamblador a máquina

```
000000 $0 $0 $t3 00000 100000
  000100 $t0 $s0 EXIT_WHILE_1
  000100 $t3 $t5 EXIT_WHILE_2
    000000 $t3 $t4 $s0 00000 100000
    000100 $t3 $s4 EXIT_WHILE_2
    000000 $t3 $s5 $t3 00000 100000
    000010 WHILE_2
  000000 $s0 $t3 $t0 00000 100000
    000010 WHILE_1
```

2. Sustuir con el diccionario de registros a binario:

000000 00000 00000 01011 00000 100000

000100 01000 10000 EXIT_WHILE_1

000100 01011 01101 BEEXIT_WHILE_2TA

000000 01011 01100 10000 00000 100000

000100 01011 10100 EXIT_WHILE_2

000000 01011 10101 01011 00000 100000

000010 WHILE_2

000000 10000 01011 01000 00000 100000

000010 WHILE_1

3. Convertir direcciones de memoria a binario

WHILE_1 232

00000000000000000011101000

WHILE_2 264

00000000000000000100001000

EXIT_WHILE_2 424

0000000110101000

EXIT_WHILE_1 488

0000000111101000

Lenguaje máquina

200 000000**00000**00000**01011**00000**100000**

232 000100**01000**10000**0000001111101000**

264 000100**01011**01101**0000001101010000**

296 000000**01011**01100**10000**00000**100000**

328 000100**01011**101000**0000001101010000**

360 000000**01011**10101**01011**00000**100000**

392 000010**0000000000000000100001000**

424 000000**10000**01011**01000**00000**100000**

456 000010**0000000000000000011101000**

Segundo examen de OAC

Valor 1, 70%; 2, 30%

Segundo examen de OAC

EJERCICIO 2

Convertir el siguiente código máquina a:

a) Ensamblador MIPS
b) Código fuente (Lenguaje C)

CÓDIGO MÁQUINA	DIRECCIÓN	FORMATO
000100**01000**10000**0000000100001000**	200	I
000000**10001**10010**01000**00000**100000**	232	R
000010**00000000000000011001000**	264	J

a) Ensamblador MIPS

```
200 WHILE: beq $t0,$s0,296
232 add $t0,$s1,$s2
264 j 200
296 EXIT:
```

Nota sobre el ensamblador:

Use el diccionario de registros del anexo.

$(264)_{10} = (0000000100001000)_2$

Valor 1, 70%; 2, 30%

Sin embargo, me parecio adecuado que termine el bucle en EXIT, a saber 296, en vez, de un bucle infinito a 200.

$(200)_{10} = (00000000000000000011001000)_2$

b) Código fuente (Lenguaje C)

a=$t0, i=$s0, j=$s1, k=$s2

```
WHILE (a!=i) {
    a=j+k;
}
```

Nombre	Campos						Comentarios
Formato	**6bits**	**5bits**	**5bits**	**5bits**	**5bits**	**6bits**	Todas las instrucciones MIPS tienen 32 bits
R	op	rs	rt	rd	shamt	funct	Formato de instrucción aritmética
I	op	rs	rt	dirección/inmediato			Formato para transferencias, saltos condicionales y op
J	op	dirección objetivo					Formato de instrucción de salto incondicional

Códigos operacionales				
add	100000		**beq**	000100
addi	001000		**bgtz**	000111
sub	100010		**blez**	000110
div	011010		**bne**	000101
mult	011000		**bge**	011111
j	000010			

Segundo examen de OAC

REGISTROS TEMPORALES							
$s0	**$s1**	**$s2**	**$s3**	**$s4**	**$s5**	**$s6**	**$s7**
16	17	18	19	20	21	22	23
$t0	**$t1**	**$t2**	**$t3**	**$t4**	**$t5**	**$t6**	**$t7**
8	9	10	11	12	13	14	15

Valor 1, 70%; 2, 30%

Anexo

Registros a bits

```
'$t0': '01000',
'$t1': '01001',
'$t2': '01010',
'$t3': '01011',
'$t4': '01100',
'$t5': '01101',
'$t6': '01110',
'$t7': '01111',
'$t8': '11000',
'$t9': '11001',
'$s0': '10000',
'$s1': '10001',
'$s2': '10010',
'$s3': '10011',
'$s4': '10100',
'$s5': '10101',
'$s6': '10110',
'$s7': '10111',
'$0': '00000'
```