

Apellido y Nombre Legajo

1.a. Traduzca la siguiente pseudo instrucción a un código equivalente en lenguaje ensamblador, pero que no utilice pseudo instrucciones:

li \$t0, 0xABC0F789	Res:
---------------------	------

1.b. Expresé el siguiente fragmento en lenguaje ensamblador MIPS. Para la construcción if then else utilice únicamente las instrucciones slt, beq, bne. largo y ancho son variable globales.

if (largo <= ancho) largo = 1000; else largo = 2000;	Res:	
	(continúe en la siguiente columna)	

1.c. Resuelva con algunas instrucciones de MIPS el siguiente requerimiento (puede utilizar pseudo instrucciones):

En la dirección de memoria principal 0x200 se encuentra una media palabra. - Si el bit 5 de esa media palabra tiene valor 1 coloque el valor 0xFFFFFFFF en la dirección 0x204. - Si el bit 5 de esa media palabra tiene valor 0 coloque el valor 0x44444444 en la dirección 0x204	Res:
---	------

1. d. Decodifique las siguientes instrucciones máquina. Indique también de qué formato es la instrucción. (ver 2da pagina)

0x8dae0004 0x01a6c024 0xa1b80000	Instrucción en ensamblador	Formato
--	----------------------------	---------

2. Una CPU MIPS cuenta con una caché para las INSTRUCCIONES MÁQUINA que ejecuta. Es decir, cada vez que la CPU debe obtener una instrucción y ejecutarla, verifica primero si está en esta caché de instrucciones. La misma es de mapeo directo de 8 entradas, y en cada entrada se almacenan 4 palabras de datos. La CPU MIPS funciona con un clock de 1Khz (mil ciclos de reloj por segundo). El acceso a caché toma 1 ciclo de reloj. El acceso a memoria principal toma 50 ciclos de reloj. Las siguientes instrucciones se encuentran en código máquina en la dirección 0x4000. Indique:

addi \$t0, \$zero, 5000 addi \$t1, \$zero, \$zero loop: beq \$t0, \$zero, salir addi \$t1, \$t1, 20 addi \$t0, \$t0, -1 j loop salir: sw \$t1, 1000(\$zero)	Cant. de accesos a memoria: Cant. de aciertos: Cant. de fallos: Tasa de aciertos: Tasa de Fallos: Tiempo medio de acceso : (con su ecuación)
---	--

Apellido y Nombre Legajo

3. Traduzca, como lo haría un compilador, la siguiente función, a lenguaje ensamblador MIPS.

```
int sumatoria (int n)
{
    unsigned char temp[100];    // cada elemento de temp ocupa un byte
    int res;  int i;

    for (i=0; i<100; i++)
        temp[i] = random(i);

    for (i=0; i<n; i++) {
        if (temp[i] != 0)
            res = res + temp[i];
    }
    return res;
}
```

Extracto del MANUAL de MIPS:

31	26	25	21	20	16	15	0
SB 101000		base		rt		offset	
6		5		5		16	
Format: SB rt, offset(base)							MIPS32 (MIPS I)

31	26	25	21	20	16	15	0
SLTI 001010		rs		rt		immediate	
6		5		5		16	
Format: SLTI rt, rs, immediate							MIPS32 (MIPS I)

31	26	25	21	20	16	15	0
LW 100011		base		rt		offset	
6		5		5		16	
Format: LW rt, offset(base)							MIPS32 (MIPS I)

31	26	25	21	20	16	15	0
ADDI 001000		rs		rt		immediate	
6		5		5		16	
Format: ADDI rt, rs, immediate							MIPS32 (MIPS I)

31	26	25	21	20	16	15	0
ORI 001101		rs		rt		immediate	
6		5		5		16	
Format: ORI rt, rs, immediate							MIPS32 (MIPS I)

31	26	25	21	20	16	15	11	10	6	5	0
SPECIAL 000000		rs		rt		rd		0 00000		AND 100100	
6		5		5		5		5		6	
Format: AND rd, rs, rt							MIPS32 (MIPS I)				

31	26	25	0								
JAL 000011		instr_index									
6		26									
Format: JAL target							MIPS32 (MIPS I)				