

| Núm. ord. | Apellido y nombre | L.U. | #hojas |
|-----------|-------------------|------|--------|
| | | | |

SISTEMAS DIGITALES - Parcial

Primer Cuatrimestre 2024

| Ej. 1 | Ej. 2 | Ej. 3 | Ej. 4 | Nota |
|-------|-------|-------|-------|------|
| | | | | |

Correctorx:

Aclaraciones

- Anote apellido, nombre, LU y numere *todas* las hojas entregadas, entregando los distintos ejercicios en hojas separadas.
- Cada ejercicio será calificado con una de las siguientes tres notas: Bien, Regular o Mal. La división de los ejercicios en incisos es meramente orientativa. Los ejercicios se calificarán globalmente.
- El parcial no es a libro abierto pero pueden utilizar la cartilla de referencia entregada por la materia.
- **Importante:** Justifique sus respuestas.
- Un resultado sin suficiente justificación equivale a un ejercicio no resuelto.
- El parcial se aprueba con al menos dos ejercicios Bien y uno Regular. Para obtener un Regular es necesario demostrar conocimientos sobre el tema del ejercicio. Para la promoción deben contar con al menos tres ejercicios bien y uno regular.

Ejercicio 1 Se cuenta con cuatro datos sin signo de un byte cada uno almacenados en el registro `s0` y queremos sumar el valor de los cuatro datos.. Escriba un programa de ensamblador RISC V que realice esta operación y almacene el resultado en el registro `a0`.

Ejemplo:

| Bits | 31 | 24 | 23 | 16 | 15 | 8 | 7 | 0 |
|-----------------|------|----|------|----|------|---|------|---|
| <code>s0</code> | 0x90 | | 0x1A | | 0x00 | | 0x02 | |

Con este dato el registro debería valer 0x000000BC.

Ejercicio 2 Implemente la función hanoi en el lenguaje ensamblador RISC V de forma recursiva, respete la convención de llamada presentada en la materia, explique el uso que le dará a cada registro y cómo se asegura que sus valores se preservan antes y después de cada llamada a función.

$$hanoi(n) = \begin{cases} 1, & \text{si } n = 1 \\ 2 * hanoi(n - 1) + 1, & \text{si } n > 1 \end{cases}$$

Guía de resolución (opcional):

- Escriba una versión de pseudocódigo.
- Transforme cada caso a su equivalente de operaciones atómicas (descomponga las operaciones lógicas, aritméticas y llamadas a función).
- Identifique los registros a emplear para cada dato.
- Si debe preservar algún registro para respetar la convención, indique qué mecanismo utilizará.
- Defina un flujo de ejecución tentativo.

Ejercicio 3 Un sistema de gestión de notas mantiene registro de las notas de una clase en un arreglo de datos de 1 byte sin signo. Queremos agregar lógica para determinar la cantidad de estudiantes que obtuvieron un valor mayor a 0xA0, que es el valor con el cual promocionan.

Se cuenta con un arreglo `notas` de datos de 8 bits sin signo empaquetados de forma contigua. El largo del arreglo (en bytes) se define en la constante `largo`.

Escriba un programa que cuente la cantidad de notas que se encuentran por sobre el valor 0xA0. Si la cantidad de valores que superan este límite es mayor a la mitad del largo debemos poner un 1 en el registro `a0`, en caso contrario debemos poner un 0.

Ejemplo:

DN7: 45238040

W: 448/23

MONTEO SUM CRU 2

① Ejercicio 1:

ADDI t0, s0, 0

ADDI t1, s0, 0

ADDI t2, s0, 0

ADDI t3, s0, 0

Carga en t0, t1, t2, t3 el valor de s0
to es memoria de 32 bits, pero se carga en
t1 " " " " 1 " " "
t2 " " " " 3 " " "
t3 " " " " 3 " " "

// Limpiar ~~valor~~ t0 para guardarlo con 16 bytes

SLLI t0, t0, 24

SRLI t0, t0, 24

// Limpiar t1 para guardarlo con byte 1

SLLI t1, t1, 16

SRLI t1, t1, 24

// Limpiar t2 para guardarlo con byte 2

SLLI t2, t2, 8

SRLI t2, t2, 24

// Limpiar t3 para guardarlo con bytes

SRLI t3, t3, 24

// Los sumo en a0

ADD a0, t0, t1

ADD a0, a0, t2

ADD a0, a0, t3

~~RET~~

RET

RET ES UNA PSEUDO INSTRUCCION PARA DAR X0
EN ESTE CONTEXTO NO ES NECESARIO HACER SO

NOTA

SVT: 45238040

LV: 44B/23

MONTEO SUM CLV2

2

② HAND PSEUDO CÓDIGO:

HAND (M):

IF (M) = -1

RET 1

USE

TEMP0 = HAND LM - 1

TEMP1 = 2 + TEMP0

TEMP2 = TEMP1 + 1

RETURN TEMP2

COMENTARIO: AO = M

Ca

EO = 1 // PARA VERIFICAR IF

VALORES A EVALUAR: Ca

EN MEMORIA

CÓDIGO RISCV

COMENTARIO: ASUMO
CONTADOR M = 1

HANDI: ADDI t0, zero, 1

BEQ a0, t0, BASE

ADDI SP, SP, -4

SW Ca, 0(SP)

ADDI a0, a0, -1

SAL HANDI

~~ADDI a0, a0, 1~~

~~ADDI a0, a0, 1~~ SLAI a0, a0, 1

ADDI a0, a0, 1

LW Ca, 0(SP)

ADDI SP, SP, 4

RET

BASE: RET

(B) ③ ~~100%~~

EXPLICACIÓN:

- ① ASIGNO 1 AL REGISTRO EO PARA VERIFICAR EL IF.
- ② VERIFICO SI M (EO) ES IGUAL A 1, SI ES IGUAL SALTO AL CASO BASE, SI NO A LA INSTRUCCIÓN SIGUIENTE $[E = M \times M \text{ PC}]$.
- ③ BUSCO GUARDAR UN ESPACIO EN MEMORIA, POR ESO RESTO 4 AL PC.
- ④ GUARDO EL VALOR DE RETORNO EN MEMORIA.
- ⑤ RESTO 1 A EO PARA ASÍ CALCULAR EL HANOI $(M-1)$.
- ⑥ HAGO UN ~~SALTO~~ SALTO A HANOI GUARDANDO EN EO EL VALOR DEL PC + 4 (GUARDA LA SIG. INSTRUCCIÓN)
- ⑦ MULTIPLICO EO POR 2 (DESPLAZO 1 BIT A IZQ) $\times 2 = \text{HANOI}(M-1)$
- ⑧ SUMO 1 A EO $\times 2 = \text{HANOI}(M-1) + 1$
- ⑨ CARGO EL VALOR DE LA MEMORIA (UN VALOR DEL ~~PRIMERO~~ ANTERIORMENTE) EN EO
- ⑩ RESTAuro / LIMPIO LA MEMORIA INCREMENTANDO EN 4
- ⑪ HAGO RET PARA SOLTAR EL VALOR DEL PC OBTENIDO CON EL SW
- ⑫ CASO BASE, SE QUE EO = 1 YA QUE CUI EN EL CASO BASE ENTONCES ME ALUNZA CON ~~EL~~ VALOR AL VALOR QUE ENTRA EN EO CON UN RET Y NO CAMBIA EL VALOR DEL 1 QUE ~~QUE~~ EN EL PSEUDO CÓDIGO QUE LA LETRAL 1



but: ysc38040
w: 448/23

MONTEO SAN CRUZ

(B-) ③ LBU

• TEXT

La a0, notas

La a1, uelco

~~And t3, zero, 0~~

LBU a1, 0(a1) // Valor de uelco sin extensión de signo

ADDI t0, a1, 0 // contador del uelco

ADDI t3, zero, 0 // contador notas < 0x A0

WHILE: BGE t0, zero, FIN

LBU t1, 0(a0)

ADDI t4, zero, 0xA0

~~SLTU~~ SLTU t2, t4, t1 // t4 es el ^{PARÁMETRO} t4 < t1 ^{NOTA} STWO 0

ADD t3, t3, t2 // sumo el valor de contador

ADDI t0, t0, -1

ADDI a0, a0, 1

J WHILE

FIN: ADDI t0, a1, 0 // me vuelvo a poner el valor del uelco

SELI t0, t0, 1

SLT a0, t0, t3 // t si uelco/2 < contador

RET

MISMO QUE EJERCICIO 1)

DNI: 45238040
L: 4/8/23

MATEO SUAN CRUZ

B

Y LA VENTAJA DE TENER UN REGISTRO CONSTANTE CERO ES EN PARTE LA FACILIDAD PARA CREAR PSEUDO INSTRUCCIONES, LOS VALORES FACILITAN AL PROGRAMADOR LA CREACIÓN DE CÓDIGO MÁS CORTO Y CONCISO. LAS VERDADERAS PSEUDO INSTRUCCIONES SON UNA CONFIGURACIÓN DE LAS INSTRUCCIONES DE DISCU CON ALGUNA CONFIGURACIÓN ESPECIAL EN LOS REGISTROS. POR EJEMPLO SI TOMAMOS RET.

ESTA INSTRUCCIÓN BÁSICAMENTE ES UNA SALIR XD, Y+1, 0 Y AL MISMO TIEMPO ES MUCHO MÁS FÁCIL DE ESCRIBIR Y RECORDAR.

ESTA CREACIÓN DE PSEUDO INSTRUCCIONES SE PERMITE YA QUE EL REGISTRO X0 NO PERMITE SU ESCRITURA Y POR ESO SI TRATO DE ESCRIBIR EN EL NO VA A AFECTAR EN NADA AL REGISTRO Y VA A SEGUIR TIENDO EL VALOR CONSTANTE "0".

OTRO PUNTO DE VISTA ES QUE ~~NO~~ RET ESTÁ USANDO EL REGISTRO X0 PARA ASIGNAR ~~CONTENIDO~~ UN REGISTRO DE DESTINO AL CUAL NO VA A PODER LUI LE INTERESA MODIFICAR, YA QUE A RET SOLO LE INTERESA SACAR EL VALOR DEL PC ALMACENADO EN RET.

LAS ESCRITURAS, POR LO DICHO ANTERIORMENTE, ESTÁN PROHIBIDAS. ES DECIR SI TO FALTO DE PISAR EL VALOR DE X0 CON OTRO VALOR SIMPLE NO VA A HACER NADA, SU VALOR SEGUIRÁ CONSTANTE EN 0.

MIENTRAS QUE LA LECTURA AL REGISTRO SI ESTÁN HABILITADAS, OSEA SI FALTO DE VER ME EL VALOR DE X0 VOY A OBTENER EL "0".

ENTENDIENDO QUE ENTENDIENDO LA IDEA, NO ES QUE ESTÉN PROHIBIDAS Y SE GANARÍA UN ERROR, SINO QUE CUANDO SE TRATA DE ESCRIBIR NO VA A TENER EFECTO ALGUNO.

ESQUEMA A EXPLICACIÓN:

DADO QUE EL DATO DE NOTAS VIENE EN BYTES ENTEROS:
PRIMERO HAGO UN FETCH DEL ~~64~~ BYTE. PARA ESO PRIMERO OBTENGO LA ADDRESS PARA ASÍ TENER LA DIRECCIÓN AL PRIMER BYTE DEL ARRAY.
LUEGO HAGO LO MISMO CON LUGO Y PARA OBTENER R LUEGO SIN SU EXTENSIÓN DE SIGNO ENTONCES HAGO LBU CON $CS1 =$ LA ADDRESS DEL LUGO.
LUEGO ME CARGO DOS CONTADORES, 1 QUE SIMULA EL RECORRIDO EN EL ARRAY (SIMULA LA CONTINUA DE EVENTOS DISTINTOS) Y OTRO QUE MANTIENE LA CANTIDAD DE NÚMEROS MAYORES A LA COTA (OXAD).

LUEGO ENTRA AL WHILE Y VERIFICA SI EL CONTADOR DE LUGO ES IGUAL A "0". SI NO ES IGUAL ENTONCES VOY A BUSCAR LUGAR EL BIT DE NOTAS EN DONDE TENGA LA ADDRESS APUNTANDO LUGO) Y LO HAGO CON LBU PARA EVITAR LA EXTENSIÓN DE SIGNO. EN LA PRÓXIMA INSTANCIACIÓN CARGO LA COTA A+1 PARA ASÍ PODER VERIFICAR SI LA COTA ES MENOR A LA NOTA, SI ESTO PASA ~~RECORRO~~ PONGO UN 1 EN ~~el contador~~, SI NO UN 0. DESPUÉS SUMO EL VALOR AL CONTADOR DE NOTAS MAYORES Y INCREMENTO LA ADDRESS EN 1 (YA QUE ME VOY A QUERER MOVER AL PRÓXIMO BYTE) Y DECREMENTO EL CONTADOR DE LUGO EN 1 (PARA ~~seguir~~ ~~contar~~ ~~los~~ ~~bits~~ ~~que~~ ~~ya~~ ~~computé~~ ~~un~~ ~~dato~~ ~~del~~ ~~array~~) Y VUELVO AL WHILE.

POR ÚLTIMO LUEGO A FIN UNA VEZ QUE NO ME QUEDAN MAS ELEMENTOS EN EL "ARRAY NOTAS" (OSEA CUANDO ME TOGA 0). AQUÍ VOY A VOLVER A TRAERME EL VALOR SIN SIGNO ALMACENADO EN EL DEL LUGO, ~~lo voy a dividir por 2~~ ^{lo voy a} ~~dividir~~ POR 2 CON UN CORRECTIVO DE UN BIT A DERECHA. ~~Y VOY A VOLVER AL COMPARAR~~

POR ÚLTIMO COMPARO Y ALMACENO EN R0 EL RESULTADO DE DIVIDIR SI EL $LUGO/2$ ES MENOR AL CONTADOR DE NOTAS MAYORES A LA COTA