

Nm. 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:

DN: 45238040

W: 448/23

MONTEO SUM CRU 2

① Ejercicio 1:

ADDI t0, t0, 0

ADDI t1, t0, 0

ADDI t2, t0, 0

ADDI t3, t0, 0

Código en t0, t1, t2, t3 El valor de 50

to a memoria el byte 0 nuevo del contador

t1 " " " 1 " " "

t2 " " " 3 " " "

// Limpio t0 para quedarme con el byte 0

SLLI t0, t0, 24

SRLI t0, t0, 24

// Limpio t1 para quedarme con byte 1

SLLI t1, t1, 16

SRLI t1, t1, 24

// Limpio t2 para quedarme con byte 2

SLLI t2, t2, 8

SRLI t2, t2, 24

// Limpio t3 para quedarme con byte 3

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 XE
EN ESTE CONTEXTO NO ES NECESARIO HACER SO

NOTA

Extinção da Colômbia.

- ④ ME COPIO EL LIMITE DE 50 EN 10, 20, 30, 40, 50
- ⑤ LIMPIEZA DE 10
DADO QUE ~~50~~ EL BYTE 0 INVOLUCRA CON LOS BITS DE 0 A 23
VOT A QUEERE QUE DIERA CON ~~50~~ EN 10.
- ⑥ Muevo todos los BITS a ~~24 bits~~ y dado que uso SLLI
ME COMPLETA CON 50 CEROS A 24. UNA VEZ QUE MOVÍ A 24 "BITS"
TENGO EL BIT "0" EN LA POSICIÓN 24, EL 1 EN LA 25 ...
Y EL 4 EN LA 31.
- ⑦ Muevo EL BYTE A SU LUGAR CORRESPONDIENTE
PARA ESO HAGO SLLI QUE ME COMPLETA ~~los bits de 0 a 23~~ con 0
Y DESPUES LOS BITS MUEVO COMO DESDE 0. POR EL PRIMER USO
TENGO QUE LUGAR A LUGAR QUE EL BIT 24 (AUSI SLLI) ESTE
EN EL BIT 0, POR LO TANTO DESPUES A DERECHA 24 BITS
- ⑧ USO LIMPIEZA BYTES: DADO QUE YA ESTA LO MAS A LA DERECHA
POSIBLE, SI LO DESPUES MUEVO INFORMACION, PARA EL CASO ESTO LO
LO DESPUES A 24, SI NO QUE A DERECHA "24 BITS" PARA
LOBLAR QUE EL BIT 24 ~~quede~~ EN EL BIT (0)
- ⑨ HAGO LA SUMA BIT A BIT ENTRE CADA UNO DE LOS REGISTROS
YA ALMACENADO EN EO.

SVT: 45238040

LV: 448/23

Montado Sun Cruz

2

② HAND PSEUDO CÓDIGO:

HAND(M):

IF ($M \neq -1$)

RET 1

USE

TEMP0 = HAND LM - 1

TEMP1 = 2 + TEMP0

TEMP2 = TEMP1 + 1

RETURN TEMP2

RELISTROS: $EO = M$

EO

EO = 1 // para verificar IF

VALORES A GUARDAR: EO

em memória

CÓDIGO RISC V

COMENTARIO: ASUMO
CONTADOR $M \geq 1$

HANDI: ADDI t0, zero, 1

BEQ a0, t0, BASE

ADDI sp, sp, -4

SW ra, 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 ra, 0(sp)

ADDI sp, sp, 4

RET

BASE: RET

(B-) ③ ~~1200~~

EXPLICACIÓN:

- ① ASIGNO 1 AL REGISTRO PC PARA VERIFICAR EL IF.
- ② VERIFICO SI M (CERO) ES IGUAL A 1, SI ES IGUAL SALTO AL CASO BASE, SI NO A LA INSTRUCCION SIGUIENTE (SUMA Y AL 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 20 PARA ASI CALCULAR EL HANOI (M-1).
- ⑥ HAGO UN ~~SALTO~~ SALTO A HANOI GUARDANDO EN EL VALOR DEL PC +4 (GUARDA LA SIG. INSTRUCCION)
- ⑦ MULTIPLICO 20 POR 2 (DESPLAZO 1 BIT A IZQ) $2 \times \text{HANOI}(M-1)$
- ⑧ SUMO 1 A 20 $2 \times \text{HANOI}(M-1) + 1$
- ⑨ CARGO EL VALOR DE LA MEMORIA (UN VALOR DEL FINICIA MENTE ANTERIORMENTE) EN EL PC
- 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 20 = 1 YA QUE CAY EN EL CASO BASE ENTONCES ME ALUNZA CON ~~EL~~ VALOR AL VALOR QUE ENTRA EL PC CON UN RET Y NO CAMBIA EL VALOR DEL PC QUE ENTRA EN EL PSEUDO CODIGO QUE LA LECTUAL 1



SVI: YS238040
W: 448/23

MONTEO SWAN CRUZ

3

(B-) ③ ~~LABA~~

• TEXT

LA a0, NOTAS

LA a1, UEGO

~~ADDI t0, zero, 0~~

LBU a1, 0(a1) // VALOR DE UEGO SIN EXTENSIÓN DE SIGNO

ADDI t0, a1, 0 // CONTADOR DEL UEGO

ADDI t3, zero, 0 // CONTADOR MAYORES A 0x A0

WHILE: BEQ t0, zero, FIN

LBU t1, 0(a0)

ADDI t4, zero, 0x A0

~~SLTU~~ SLTU t2, t4, t1 // t4 < t1 ^{PARÁMETRO} _{NOTA} STNO 0

ADD t3, t3, t2 // SUMO EL VALOR A2 CONTADOR

ADDI t0, t0, -1

ADDI a0, a0, 1

J WHILE

FIN: ADDI t0, a1, 0 // NO VUOLVO A RECIBIR EL VALOR DEL UEGO

SELI t0, t0, 1

SLT a0, t0, t3 // 1 SI UEGO/2 < CONTADOR

RET MISTO (LOS EJERCICIO 1)

NOTA

DNI: 45238040

W: 448123

MONTAÑA SUAN CRUZ

Página 4

B

① La ventaja de tener un registro constante como es el PC, la facilidad para crear pseudo instrucciones, las cuales se facilitan al programador la creación de código más corto y conciso. Las ~~ventajas~~ pseudo instrucciones son una configuración de las instrucciones de RISC-V con alguna configuración especial en los registros. Por ejemplo si tomamos RET.

Esta instrucción básicamente es una salte $x0, x1, 0$ y al mismo tiempo es mucho más fácil de escribir y recordar.

Esta creación de pseudo instrucciones se permiten ya que el registro $x0$ no permite su escritura y por eso si intentamos escribir en él no va a afectar en nada al registro y va a seguir siendo el valor constante "0".

Otro punto de vista es que ~~por~~ RET está usando el registro $x0$ para asignar ~~constantemente~~ un registro de destino al cual no va a poder ~~un lo interesa~~ modificar, ya que a RET solo le interesa saltar al valor del PC almacenado en él.

② Las escrituras, por lo dicho anteriormente, están prohibidas. Es decir si intentamos escribir 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, o sea si intentamos leer el valor de $x0$ voy a obtener el "0".

Entiendo que entiendo la idea, no es que estén prohibidas y se genere un error, sino que cuando se trata de escribir no va a tener efecto alguno.

ESERCIZIO 2 ESPERACION:

DADO QUE EL DATO DE NOTAS VIENE EN BYTES ENTONCES:
PRIMERO HAGO UN FETCH DEL ~~8~~ BYTE. PARA ESO PRIMERO OBTENGO
LA ADDRESS PARA ASI TENER LA DIRECCION AL PRIMER BYTE DEL ARRAY.
LUEGO HAGO LO MISMO CON LECO Y PARA OBTENER EL LECO SIN
SU EXTENSION DE SIGNO ENTONCES HAGO LBU CON EST = LA ADDRESS DEL LECO.
LUEGO ME LLEGO DOS CONTADORES, 1 QUE SIMULA EL RECORRIDO EN EL ARRAY
(SIMULA LA CANTIDAD DE EVENTOS DESTINATOS) Y OTRO QUE MANTIENE LA
CANTIDAD DE NUMEROS MAYORES A LA COTA (OXFLO)

LUEGO ENTRA AL WHILE Y VERIFICO SI EL CONTADOR DE LECO ES IGUAL
A "0". SI NO ES IGUAL ENTONCES VOY A BUSCAR CARGAR EL BIT DE NOTAS
EN DONDE TENGA LA ADDRESS APUNTANDO LECO) Y LO HAGO CON LBU PARA EVITAR
LA EXTENSION DE SIGNO. EN LA PROXIMA INSTANCIACION CARGO LA COTA A+4
PARA ASI PODER VERIFICAR SI LA COTA ES MENOR A LA NOTA,
SI ESTO PASA ~~pongo~~ PONGO UN 1 EN ~~la~~ ^{el} ~~contador~~, SI NO UN 0. DESPUES SUMO EL
VALOR AL CONTADOR DE NOTAS MAYORES Y INCREMENTO LA ADDRESS EN 1
BYT QUE ME VOY A QUERER MOVER AL PROXIMO BYTE) Y DECREMENTO
EL CONTADOR DE LECO EN 1 (PARA ~~decrementar~~ ^{decrementar} ~~que ya~~ ^{que ya} COMPUTE UN DATO DEL ARRAY
Y VUELVO AL WHILE.

POR ULTIMO LLEGO A FIN UNA VEZ QUE NO ME QUEDAN MAS ELEMENTOS
EN EL "ARRAY NOTAS" (OSEA CUANDO MI ~~sea~~ ^{sea} 0). AGUI VOY A VOLVER
A TRAERME EL VALOR SIN SIGNO ALMACENADO EN el del LECO, ^{lo} VOY A
DIVIDIR POR 2 CON UN CORRECTIVO DE UN BIT A DER. ~~VOY A~~
~~ALMACENAR~~

POR ULTIMO COMPARO Y ALMACENO EN el EL RESULTADO DE DIVISION
SI EL LECO/2 ES MENOR AL CONTADOR DE NOTAS MAYORES A LA COTA

NOTA

NOTA