Pontificia Universidad Católica de Chile
Escuela de Ingeniería
Departamento de Ciencia de la Computación

IIC2343 — Arquitectura de Computadores — 1' 2022

Interrogación 2

Responde solo TRES de las siguientes cuatro preguntas.

1 Cache

Considere una cache de 16KiB y palabras de 4 bytes. Se tienen los siguientes accesos:

1. `0xA590`
2. `0xA093`
3. `0x249E`
4. `0x4595`
5. `0x86B8`
6. `0x72B2`
7. `0x6C94`
8. `0xA59F`
9. `0x8EB3`
10. `0x92B5`
11. `0x2492`
12. `0x86BC`
13. `0x37FF`
14. `0x0AB2`
15. `0x47F8`
16. `0x72BA`

(a) Considere que la cache esta dividida en lineas de 16 palabras, con función de correspondencia *direct mapping*. Muestre como se divide una dirección cualquiera entre tag, índice y offset. También indique cual es el *hit-rate* y en que accesos ocurren. **(2 pts)**

(b) Considere que la cache esta dividida en lineas de 64 palabras, con función de correspondencia *direct mapping*. Muestre como se divide una dirección cualquiera entre tag, índice y offset. También indique cual es el *hit-rate* y en que accesos ocurren. **(2 pts)**

(c) Considere que la cache esta divida en lineas de 16 palabras, con función de correspondencia *4-way associative* y política de reemplazo LRU. Para cada dirección muestre el estado del set correspondiente. Indique cual es el hit-rate y en que accesos ocurren. **(2 pts)**

2 Memoria Virtual

Se tiene un computador con memoria RAM de 64 KiB, sin embargo, los programas corren con una memoria virtual de 2 MiB. Además, se sabe que los *page frames* son de 8 KiB y sus direcciones de memoria virtual van desde `0x00000` a la `0xFFFFF`.

Por otra parte, ustedes saben que su computador emplea tablas de paginas en donde cada entrada almacena 1 *bit* de presencia, 1 *dirty bit*, *m bits* suficientes para indicar el marco y todos los bits de metadata necesarios para el criterio de reemplazo, el cual sera LRU. Pueden suponer que las tablas de paginas se almacenan 'mágicamente' en otra memoria aparte, con suficiente espacio para que todo quepa correctamente.

Finalmente, las especificaciones de su computador indican que este cuenta con un con una unidad TLB, capaz de almacenar tan solo 4 traducciones. El criterio de reemplazo para esta es LRU.

Considere que tanto la tabla de paginas y la TLB comienzan vacías. En base, a la información anterior y la siguiente secuencia de accesos a memoria (direcciones virtuales) indique:

(a) Cantidad de *page frames* de la memoria física **(1 pts)**

(b) Cantidad de paginas de la memoria virtual **(1 pts)**

(c) Cantidad de *page faults* **(2 pts)**

(d) *Hit-rate* de la TLB **(2 pts)**

Accesos a memoria:

1. `0xE4021`
2. `0x12C98`
3. `0xE391D`
4. `0x01071`
5. `0x412AA`
6. `0x129E1`
7. `0x341C1`
8. `0xE4F56`
9. `0x00E46`
10. `0x018D1`
11. `0x45810`
12. `0x41EE1`
13. `0x67191`
14. `0x34E11`
15. `0xE3129`
16. `0xE4819`

3 Pipelining

Para realizar esta pregunta se tiene el pipeline de 5 etapas del computador básico visto en clases.

(a) Para el siguiente trozo de código, identifique los hazards producidos y soluciónelos utilizando NOPs, junto con indicar el hardware que lo maneja. Suponga que A está inicializado en 2 y B está inicializado en 0. **(3 pts)**

```
1       lab:
2           ADD A, 2
3           ADD B, A
4           ADD A, 3
5           CMP A, B
6           JLT lab
7       MOV (var1), A
8       MOV A, (var2)
9       MOV B, (var3)
10      ADD A, 4
11      ADD A, B
```

(b) Suponga que ejecuta el siguiente trozo de código en el computador con pipeline, pero que no se toma ninguna precaución para evitar los hazards de datos. ¿Cuál sería el valor final de A, B y el valor al que apunta la dirección `var` , dado que A y B parten inicializados en 0 y 1? **(2 pts)**

```
1       ADD A, 5
2       ADD A, B
3       MOV (var), A
```

(c) Describa qué debe ocurrir con el pipeline cuando la CPU recibe una interrupción de I/O que debe ser atendida. **(1 pts)**

4 I/O

(a) Modifique el computador básico del diagrama adjunto, para que este soporte **un** dispositivo de I/O mapeado por memoria, con soporte para interrupciones. Dibuje y describa las modificaciones necesarias, junto con las instrucciones, señales de control y opcodes que sea necesario agregar o modificar. **(2 pts)**

(b) Asumiendo que la parte a) es correcta, considere todo el *hardware* necesario para dar soporte a $N+1$ dispositivos con 3 niveles de prioridad diferentes. Describa el hardware necesario, junto con las instrucciones, señales de control y opcodes que sea necesario agregar o modificar. No es necesario que dibuje un nuevo diagrama o anote sobre la parte a) si la respondió, pero puede hacerlo si lo considera necesario. **(2 pts)**

(c) La placa Diligent Nexys-4 posee, entre varios dispositivos MMIO, un *display* de 8 dígitos, mapeado a la dirección de memoria `0x00007f18`, donde escribir a esta dirección hará que el *display* muestre en hexadecimal la palabra almacenada en dicha dirección; 5 botones mapeados a la dirección `0x00007f24`, donde los primeros 5 bits de la palabra almacenada corresponden al estado de cada botón (1 para presionado, 0 si está suelto); y un timer, que mide el tiempo que la placa lleva encendida, mapeado a la dirección `0x00007f30`, leer de esta dirección entrega el tiempo, escribir a dicha dirección reinicia el timer al valor escrito. **Escriba en assembly RISC-V una subrutina** llamada `isr_disp`, que si detecta que está presionado el botón 3, escribe el valor del timer al *display* de 7 segmentos, y en caso de ser cualquier otro botón, reinicia el timer a 0. **(2 pts)**

Decimal	Hexadecimal	Binario
0	0x0	0000
1	0x1	0001
2	0x2	0010
3	0x3	0011
4	0x4	0100
5	0x5	0101
6	0x6	0110
8	0x8	1000
9	0x9	1001
10	0xA	1010
11	0xB	1011
12	0xC	1100
13	0xD	1101
14	0xE	1110
15	0xF	1111

Considerar los prefijos:
$2^{10} = 1\text{KiB}$ $2^{20} = 1\text{MiB}$ $2^{30} = 1\text{GiB}$ $2^{40} = 1\text{TiB}$

Basic RV32I instructions, pseudo instructions and calling convention.

add t1,t2,t3 Addition: Set t1 to (t2 plus t3)

addi t1,t2,imm Addition immediate: Set t1 to t2

and t1,t2,t3 Bitwise AND: Set t1 to t2 AND t3

andi t1,t2,imm Bitwise AND imm: t1 = t2 AND imm bitwise AND of t2 and sign-extended 12-bit immediate

beq t1,t2,label Branch if equal

bge t1,t2,label Branch if greater than or equal

bgeu t1,t2,label ‘’(unsigned)

blt t1,t2,label Branch if less than

bltu t1,t2,label ‘’(unsigned)

bne t1,t2,label Branch if not equal

call label CALL: call a far-away subroutine

div t1,t2,t3 Division: set t1 to t2/t3

divu t1,t2,t3 ‘’(unsigned)

j label Jump : Jump to statement at label

jal label Jump And Link: Jump to statement at label and set the return address to ra

jal t1, target Jump and link : Set t1 to Program Counter (return address) then jump to target address

jr t0 Jump Reg: Jump to address in t0

jalr t1, t2, imm Jump and link register: Set t1 to Program Counter (ra) then jump to t2 + immediate

la t1,label Load Address : Set t1 to label

lb t1, -100(t2) Set t1 to sign-extended 8-bit value from effective memory byte address

lbu t1, -100(t2) ‘’ (unsigned)

li t1,imm Load Immediate : Set t1 to 12-bit immediate (sign-extended)

lw t1, -100(t2) Set t1 to contents of effective memory word address

lwu lw t1, -100(t2) ‘’ (unsigned)

mul t1,t2,t3 Multiplication: set t1 to t2*t3

mret Return from interrupt or exception in M-mode (to uepc)

mv t1,t2 MoVe : Set t1 to contents of t2

not t1,t2 Bitwise NOT (bit inversion)

or t1,t2,t3 Bitwise OR : Set t1 t2 OR t3

ori t1,t2,imm Bitwise OR: Set t1 to t2 OR -100

rem t1,t2,t3 Remainder: set t1 to rem of t2/t3

sll t1,t2,t3 Shift left: Set t1 to result of shifting t2 left by number of bits specified by value in low-order 5 bits of t3

slli t1,t2,imm ‘’shifting t2 left by number of bits specified by imm

slliw t1,t2,10 Shift left logical (32 bit)

sllw t1,t2,t3 Shift left logical (32 bit)

slt t1,t2,t3 Set less than: t2 is less than t3, then set t1 to 1 else set t1 to 0

slti t1,t2,imm ‘’

sra t1,t2,t3 Shift right arithmetic: Set t1 to result of sign-extended shifting t2 right by number of bits specified by value in low-order 5 bits of t3

srai t1,t2,imm ‘’shifting t2 right by number of bits specified by immediate

sraiw t1,t2,imm Shift right arithmetic (32 bit)

sraw t1,t2,t3 Shift left logical (32 bit): Set t1 to result of shifting t2 left by number of bits specified by value in low-order 5 bits of t3

srl t1,t2,t3 Shift right logical: Set t1 to result of shifting t2 right by number of bits specified by value in low-order 5 bits of t3

srli t1,t2,33 ‘’ shifting t2 right by number of bits specified by immediate

srliw t1,t2,10 Shift right logical (32 bit

srlw t1,t2,t3 Shift left logical (32 bit)

sub t1,t2,t3 Subtraction: set t1 to t2-t3

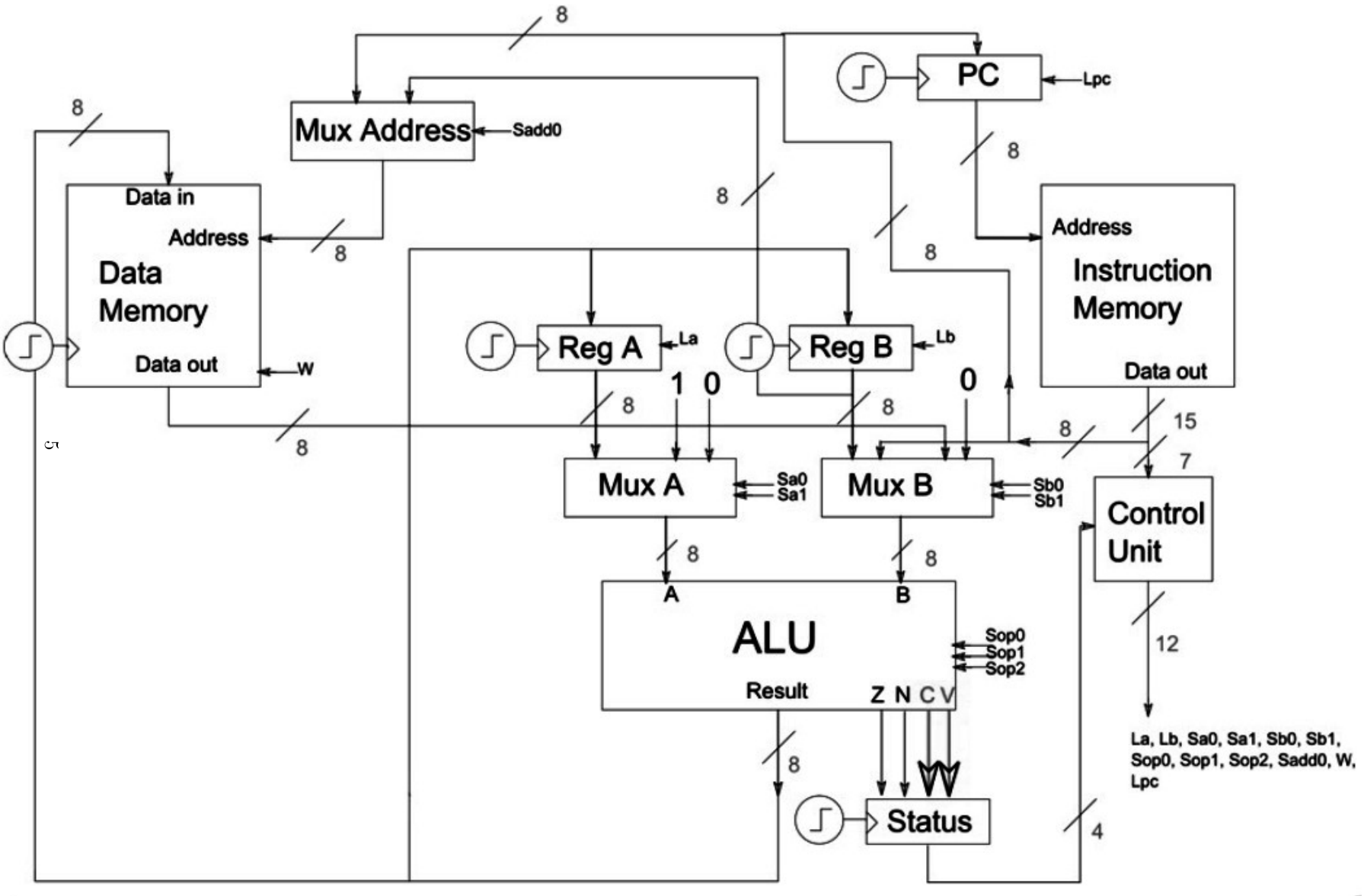
sw t1, -100(t2) Store word: Store contents of t1 into effective memory word address

xor t1,t2,t3 Bitwise XOR : Set t1 t2 XOR t3

uret Return from interrupt or exception in U-Mode (to uepc)

xori t1,t2,imm ‘’t2 XOR imm(12-bit sign extend)

Name	Number	Use
zero	x0	Constant 0
ra	x1	Return address
sp	x2	Stack pointer
gp	x3	Global pointer
tp	x4	Thread pointer
t0-2	x5-7	Temporary registers
s0/fp	x8	Saved register/frame pointer
s1	x9	Saved register
a0-1	x10-11	Arguments/return values
a2-7	x12-17	Function arguments
s2-11	x18-27	Saved registers
t3-6	x28-31	Temporary registers


8
PC
Lpc
Mux Address
Sadd0
8
Data in
Address
Data
Memory
Data out
W
8
8
8
Address
Instruction
Memory
Data out
Reg A
La
Reg B
Lb
1 0
0
8
8
8
15
7
Mux A
Sa0
Sa1
Mux B
Sb0
Sb1
Control
Unit
8
8
A
B
ALU
Sop0
Sop1
Sop2
Result
Z N C V
12
La, Lb, Sa0, Sa1, Sb0, Sb1,
Sop0, Sop1, Sop2, Sadd0, W,
Lpc
8
Status
4