CMSC411 Quiz 15

Your email will be recorded when you submit this form

* Required

What operation copies data from main memory into a general purpose register? *

● load

○ store

○ move

○ add

Which one of the following addresses is word aligned? *

○ 0x01234567

● 0x00FA0700

○ 0x77000003

○ 0x00000042

Say that four bytes in main storage contain bits that represent a 32-bit integer. What is the address is used for this integer? *

○ The address of the byte with the highest address of the four.

○ The address of each byte is used.

○ Only one byte at a time can be addressed.

● The address of the byte with the lowest address of the four.

Here is a 32-bit pattern: 0x00224477. This pattern is to be stored in main memory using bytes at addresses 0x10000000, 0x10000001, 0x10000002, and 0x10000003. On a big endian processor, what bit pattern is contained in address 0x10000000? *

● 0x00

○ 0x22

○ 0x44

○ 0x77

A lw is to load register $5 from location 0x0040000C in memory. Register $10 contains 0x00400000. Write the assembly language instruction: *

○ lw $10,0x0C($10)

○ lw $10,0x0C($5)

● lw $5,0x0C($10)

○ lw $5,0x0C(400000)

Register $10 contains 0x10000000. Beginning at that address there are five integers in a row. Write the instruction that loads the last integer into register $7. *

○ lw $7, 50($10)

○ lw $7, 20($10)

● lw $7, 16($10)

○ lw $7, 40($10)

Register $5 contains the address 0x10000100. Write the instruction that loads the four bytes that precede this address into register $7. *

○ lw $7, 4($5)

● lw $7, -4($5)

○ lw $5, 4(-$5)

○ lw $7, 0($5-4)

Write the assembly instruction that fills register $10 with 0x10000000 *

● lui $10, 0x1000

○ lui $10, 0x10000000

○ ori $10, $0, 0x10000000

○ ori $10, $10, 0x1000

Say that somehow register $10 has been loaded with the address 0x10000000. Write the instruction that alters $10 so that it contains 0x100000F0. *

○ ori $10, $0, 0x00F0

○ or $10, $10, 0x00F0

● ori $10, $10, 0x00F0

○ andi $10, $10, 0x00F0

Examine the following program fragment.   Assuming that SPIM starts the data section at address 0x10000000, what address does symbolic address result represent? *

```
## fragment.asm

        .text
        .... program statements ...

        .data

value:  .word 23
result: .word 97
```

● 0x10000004

○ 0x10000000

○ 0x10000003

○ 0x00000097

Refer back to the previous fragment. We want register $8 to contain the address of the first byte in the data section. What instruction does this? *

○ lui $8,1000

○ lui $8,$8,0x1000

○ lui $8,0x10000000

● lui $8,0x1000

Refer back to the previous fragment. Assume that register $8 contains the address 0x10000000. What instruction stores the contents of register $4 into location result? *

● sw $4,4($8)

○ sw $4,0($8)

○ sw $4,0x0($8)

○ sw $4,$0x05($8)

Get link

CMSC411 Quiz 16

Your email will be recorded when you submit this form

* Required

What is the smallest addressable unit of main memory? *

● byte

○ bit

○ nibble

○ halfword

Which of the following instructions does sign extension? *

○ lbu

● lb

○ add

○ lhu

Say that: What is *
put in register $8 after lb $8,0($5) is executed?

- Memory at `0x10000000` contains `0x80`
- Register $5 contains `0x10000000`

○ 0x88888880

○ 0x00000080

○ 0x80000000

● 0xFFFFFF80

What instruction is used to store a byte to memory? *

● sb

○ sbu

○ lb

○ sw

Say that the MIPS chip is running in little-endian mode (as does SPIM on an Intel computer). What is put in register $8 after lh $8,0($5) is executed? *

- Memory at `0x10000000` contains `0x80`
- Memory at `0x10000001` contains `0x00`
- Register $5 contains `0x10000000`

○ 0xFFFFFF80

○ 0x88888880

● 0x00000080

○ 0x80000000

Say that the MIPS chip is running in big-endian mode (as does SPIM on an Apple computer). What is put in register $8 after lh $8,0($5) is executed? *

- Memory at `0x10000000` contains `0x80`
- Memory at `0x10000001` contains `0x00`
- Register $5 contains `0x10000000`

● 0xFFFF8000

○ 0xFFFFFF80

○ 0x00000080

○ 0x80000000

Which one of the following address are half-word aligned? *

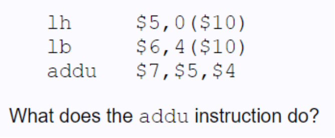
○ 0x01004F35

○ 0x01004F37

● 0x01004F3A

○ 0x01004F3F

Say that data is in memory and the base register has been initialized correctly. You have the following program: *



```
lh      $5,0($10)
lb      $6,4($10)
addu    $7,$5,$4
```

What does the `addu` instruction do?

● It performs the binary addition algorithm on whatever 32-bit patterns are in registers $4 and $5.

○ It performs a 16-bit addition because that is the size of the largest operand.

○ It performs an 8-bit addition.

○ The instruction causes a trap because the operands are not the same sizes.

Which of the following assembler directives reserves 1210 bytes of memory? *

○ .word 3

○ .byte 12

○ .block 6

● .space 12

You wish to speed up the execution of a C program. The program runs on a 32-bit processor. You notice that the variables in the program are a mix of short int, int and long int variables. The program does a great deal of integer arithmetic. How might you speed up this program? *

● Make as many variables of type int as is possible.

○ Make as many variables of type short int as is possible.

○ Make all variables as small as is needed for the range of values they are exected to hold.

○ Shorten the names of all the variables.

A digital image is stored in a file. The pixels of the image represent a gray level of 0 to 255. What instruction are you likely to use in loading a register with the value of a pixel? *

○ lb

● lbu

○ lh

○ lhu

How does SPIM display the data section of simulated main memory? *

○ One byte per address in columns.

○ In groups of 4-byte words with the highest address on the right.

● In groups of 4-byte words with the lowest address on the right.

○ This depends on the type of data in memory.

Get link

CMSC411 Quiz 17

Your email will be recorded when you submit this form

* Required

What are the three steps in the machine cycle? *

○ increment the PC; fetch the instruction; execute the instruction

○ fetch the instruction; execute the instruction; increment the PC

○ execute the instruction; fetch the instruction; increment the PC

● fetch the instruction; increment the PC; execute the instruction

What are the four bytes immediately following a jump instruction called? *

○ fetch delay slot

○ pipeline delay slot

● branch delay slot

○ PC advance slot

What is a pipeline? *

○ Several words of data from memory are moved into the processor before instructions need them.

● Several sequential instructions are simultaneously prepared for execution while one instruction finishes its execution.

○ A single instruction is divided into four phases and each phase is executed in one machine cycle.

○ Multiple items of data are sent down the system bus like water in a pipe.

Say that a sll instruction is located in memory at address 0x400100, and an add instruction is located in memory at address 0x400104. After the add instruction executes, what value will be in the PC? *

○ 0x400100

○ 0x400104

○ 0x400105

● 0x400108

Say that a j (jump) instruction is located in memory at address 0x400100, and a sll instruction is located in memory at address 0x400104. After the j instruction executes, what value will be in the PC? *

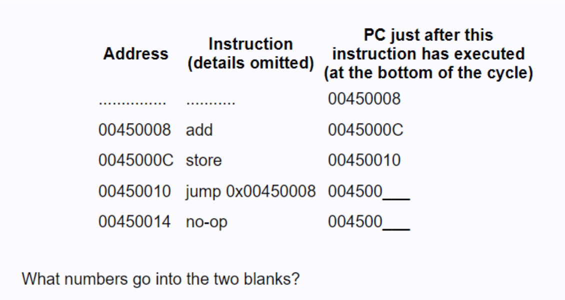
○ 0x400100

○ 0x400101

○ 0x400102

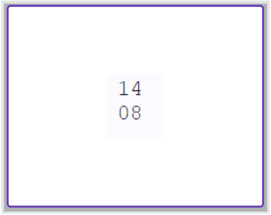
● 0x400104

Here is a schematic program loop. *



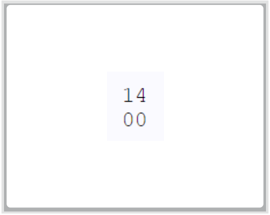
Address	Instruction (details omitted)	PC just after this instruction has executed (at the bottom of the cycle)
....	....	00450008
00450008	add	0045000C
0045000C	store	00450010
00450010	jump 0x00450008	004500___
00450014	no-op	004500___

What numbers go into the two blanks?



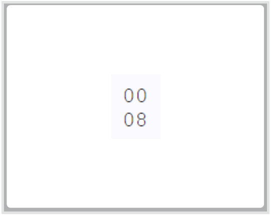
14
08

● a



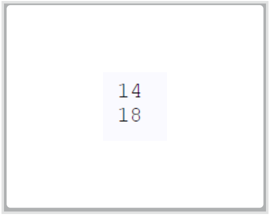
14
00

○ b



00
08

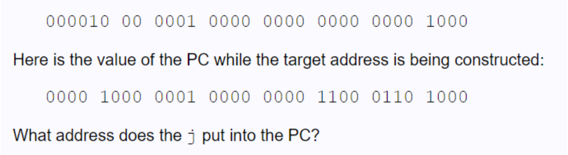
○ c



14
18

○ d

Here is a 32-bit j instruction. The first 6 bits are the op-code. *



```
    000010 00 0001 0000 0000 0000 0000 1000
```

Here is the value of the PC while the target address is being constructed:

```
    0000 1000 0001 0000 0000 1100 0110 1000
```

What address does the `j` put into the PC?

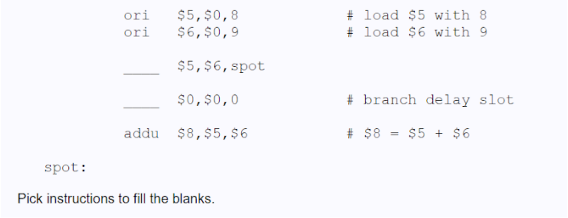
● 0000 00 0001 0000 0000 0000 0000 1000 00

○ 0000 1000 0001 0000 0000 1100 0110 1000

○ 0000 10 0001 0000 0000 1100 0110 1000 00

○ 1000 00 0001 0000 0000 0000 0000 1000 00

Examine the following program fragment. The program is to add $5 and $6 together only if they are not equal. *



```
          ori   $5,$0,8          # load $5 with 8
          ori   $6,$0,9          # load $6 with 9

          ____  $5,$6,spot

          ____  $0,$0,0          # branch delay slot

          addu  $8,$5,$6         # $8 = $5 + $6

    spot:
```

Pick instructions to fill the blanks.

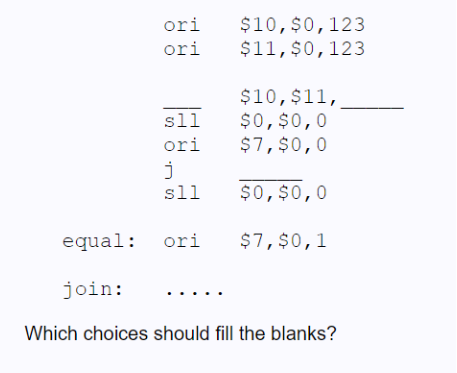
○ beq ; addu

○ bne ; sll

○ bne ; addu

● beq ; sll

Here is an if-then-else structure. The code is to compare $10 and $11. If these registers contain the same bit pattern, set register $7 to 1. Otherwise set $7 to 0. *



```
           ori   $10,$0,123
           ori   $11,$0,123

           ___   $10,$11,_____
           sll   $0,$0,0
           ori   $7,$0,0
           j     _____
           sll   $0,$0,0

    equal: ori   $7,$0,1

    join:  .....
```

Which choices should fill the blanks?

○ bne ; equal ; join

○ beq ; join ; equal

● beq ; equal ; join

○ bne ; join ; equal

Say that registers $5 and $6 each contain an ASCII character in the low order byte. Can the beq instruction be used to compare the characters? *

○ Yes, because beq will recognize the character data and do a character comparison.

○ No, because beq only works with two's complement integers.

○ No, because beq only works with full 32-bit data.

● Yes, because beq compares bit patterns regardless of what they represent.

CMSC411 Quiz 18

Your email will be recorded when you submit this form

* Required

Examine the following program fragment: *

```
          ori   $8,$0,13
          ori   $9,$0,1
          bltz  $8,target
          sll   $0,$0,0
          ori   $9,$0,0

  target: sll   $0,$0,0       # arbitrary instruction
```

What value is found in $9 when control reaches *target*?

● 0

○ 1

○ 4

○ 13

Trick Question: Examine the following program fragment: *

```
          ori   $8,$0,-57
          ori   $9,$0,1
          bltz  $8,target
          ori   $9,$0,0       # think about the delay
                              # slot
  target: sll   $0,$0,0       # arbitrary instruction
```

What value is found in $9 when control reaches *target*?

● 0

○ 1

○ 3

○ 4

Examine the following program fragment: *

```
          ori   $8,$0,13
          ori   $9,$0,1
          bgez  $8,target
          sll   $0,$0,0
          ori   $9,$0,0

  target: sll   $0,$0,0       # arbitrary instruction
```

What value is found in $9 when control reaches *target*?

○ 0

● 1

○ 4

○ 13

Examine the following program fragment (slightly different from the previous): *

```
          ori   $8,$0,13
          bgez  $8,target
          ori   $9,$0,1
          ori   $9,$0,0

  target: sll   $0,$0,0       # arbitrary instruction
```

What value is found in $9 when control reaches *target*?

○ 0

● 1

○ 4

○ 13

Examine the following program fragment: *

```
            addiu    $3,$0,-13
            addiu    $7,$0,23
            ?????
```

Pick the instruction to replace ????? that will set register $10 to one.

○ sltu $3,$7,$10

○ slt $10,$7,$3

● slt $10,$3,$7

○ sltu $10,$3,$7

Examine the following program fragment: *

```
            addiu     $3,$0,-13
            slti      $5,$3,-8
```

What value is in $5 after both instructions exectute?

○ 0

● 1

○ -8

○ -13

Examine the following program fragment: *

```
            ori       $3,$0,25
            slti      $5,$3,53
```

What value is in $5 after both instructions exectute?

○ 0

● 1

○ 25

○ 53

(Very Tricky:) Examine the following program fragment: What value is in $5 after both instructions exectute? *

```
    addiu     $3,$0,-1
    slti      $5,$3,17
```

● 0

○ 1

○ -8

○ -13

Which style of implementing a counting loop is usually easiest to understand?

○ data driven loop

○ bottom driven loop

○ conditional driven

● top driven loop

Examine the following program fragment: *

```
          ori    $5,$0,5        # initialize count
          ori    $8,$0,0        # initialize accumulator

   test:  bltz   $5,done
          sll    $0,$0,0
          addu   $8,$8,$5       # add count to accumulator
          addiu  $5,$5,-1
          j      test
          sll    $0,$0,0

   done:  sll    $0,$0,0
```

How many times is the addu instruction executed?

○ 0

○ 5

● 6

○ 7

Get link

This form was created inside of UMBC. Report Abuse

Google Forms