Brandon Cortez

CprE 381

HW 5

10/4/2020

1. Application Benchmarking

   a. The code returns the number of identical elements between the array in $v0

```
        sll $a2, $a2, 2          #Stores the value of $a2 shifted left 2 bits ($a2*4) filled by zeros back into $a2
        sll $a3, $a3, 2          #Stores the value of $a3 shifted left 2 bits ($a3*4) filled by zeros back into $a3
        add $v0, $zero, $zero    #Set $v0 to 0
        add $t0, $zero, $zero    #Set $t0 to 0
Outer:  add $t4, $a0, $t0        #Set $t4 to the address of $a0[$t0]
        lw $t4, 0($t4)           #Load the value of $a0[$t0] (array 1[i])
        add $t1, $zero, $zero    #Set $t1 to 0
Inner:  add $t3, $a1, $t1        #Set $t3 to the address of $a1[$t1]
        lw $t3, 0($t3)           #Load the value of $a1[$t1] (array 2[j])
        bne $t3, $t4, Skip       #if 1[i] != 2[j] (values are not equivalent) skip
        addi $v0, $v0, 1         #Increment $v0 by 1
Skip:   addi $t1, $t1, 4         #Set $t1 to $t1+4 (next value)
        bne $t1, $a3, Inner      #if $t1 != 2500*4 go to inner
        addi $t0, $t0, 4         #Set $t0 to $t0+4 (next value)
        bne $t0, $a2, Outer      #if $t0 != 2500*4 go to outer
```

   b. 4 + (2500 * (1+2+1+1+2)) + (2500 * 2500 * (1+2+2+1+1+2)) cycles

      4 + (2500 * 7) + (2500 * 2500 * 9) cycles

      56267504 cycles

      56267504 cycles / 2 GHz (2*10^9)

      28 ms

2. MIPS SIMD Programming

a. SAD Procedure

```
        addu $v1, $zero, $zero   #Set sum to 0
        addu $t0, $zero, $zero   #Set i to 0
        addu $t1, $zero, $zero   #Set diff to 0
        addiu $t4, $zero, 8      #Set byte overflow length
Loop:   addu $t2, $a0, $t0       #Set $t2 to the address of $a0[$t0]
        lb $t2, 0($t2)          #Load the value of array 1[i]
        addu $t3, $a1, $t0       #Set $t3 to the address of $a1[$t0]
        lb $t3, 0($t3)          #Load the value of array 2[i]
        subu $t1, $t2, $t3       #Set $t1 fo the difference between $t2 and $t3
        bgt $t1, $zero, Pos     #If diff is positive skip the sign switch step
        subu $t1, $zero, $t1     #Subtract the diff from zero to get the positive equivalent
Pos:    addu $v1, $v1, $t1       #Add the difference to the sum of differences
        addiu $t0, $t0, 1        #Increment $t0 by 1
        beq $t0, $t4, Exit      #If i = 8, more than a byte long exit
        bne $t0, $a2, Loop      #If i != array len Loop
Exit:   jr $ra                  #Jump to $ra
```

b. SAD Triple Call

```
.data
prompt: .asciiz "\n\nThe Absolute Sum of Differences is "
test1arr1: .byte 1, 1, 1, 1, 1, 1, 1, 1, 1
test1arr2: .byte 0, 0, 0, 0, 0, 0, 0, 0, 0
size1: .word 9

test2arr1: .byte 1, 2, 1, 1, 1, 1
test2arr2: .byte 0, 2, 0, 1, 1, 0
size2: .word 6

test3arr1: .byte 1, 2, 0, 1
test3arr2: .byte 3, 1, 1, 0
size3: .word 4


.text
.globl main
main:

         addi $s0, $zero, 1
Trial1:  la     $a0, test1arr1
         la     $a1, test1arr2
         lw     $a2, size1
         j Test

Trial2:  la     $a0, test2arr1
         la     $a1, test2arr2
         lw     $a2, size2
         j Test

Trial3:  la     $a0, test3arr1
         la     $a1, test3arr2
         lw     $a2, size3
         j Test

Test:   addi $s0, $s0, 1        #increment the test counter
        jal     func            # Save current PC in $ra, and jump to func
```

```
Test:   addi $s0, $s0, 1        #increment the test counter
        jal     func            # Save current PC in $ra, and jump to func

        la $a0, prompt
        li $v0, 4
        syscall

        addi $a0, $v1, 0
        li  $v0, 1
        syscall         #Output sum of absolute differences of the test

        beq $s0, 2, Trial2
        beq $s0, 3, Trial3
        li  $v0, 10
        syscall         #Exit


func:   #Compute sum of absolute differences
        addu $v1, $zero, $zero   #Set sum to 0
        addu $t0, $zero, $zero   #Set i to 0
        addu $t1, $zero, $zero   #Set diff to 0
        addiu $t4, $zero, 8      #Set byte overflow length
Loop:   addu $t2, $a0, $t0       #Set $t2 to the address of $a0[$t0]
        lb $t2, 0($t2)          #Load the value of array 1[i]
        addu $t3, $a1, $t0       #Set $t3 to the address of $a1[$t0]
        lb $t3, 0($t3)          #Load the value of array 2[i]
        subu $t1, $t2, $t3       #Set $t1 fo the difference between $t2 and $t3
        bgt $t1, $zero, Pos     #If diff is positive skip the sign switch step
        subu $t1, $zero, $t1     #Subtract the diff from zero to get the positive equivalent
Pos:    addu $v1, $v1, $t1       #Add the difference to the sum of differences
        addiu $t0, $t0, 1        #Increment $t0 by 1
        beq $t0, $t4, Exit      #If i = 8, more than a byte long exit
        bne $t0, $a2, Loop      #If i != array len Loop
Exit:   jr $ra                  #Jump to $ra
```

c. Quad Byte Form

```
.data
prompt: .asciiz "\n\nThe Absolute Sum of Differences is "
test1arr1: .byte 1, 1, 1, 1, 1, 1, 1, 1, 1
test1arr2: .byte 0, 0, 0, 0, 0, 0, 0, 0, 0
size1: .word 9

test2arr1: .byte 1, 2, 1, 1, 1, 1
test2arr2: .byte 0, 2, 0, 1, 1, 0
size2: .word 6

test3arr1: .byte 1, 2, 0, 1
test3arr2: .byte 3, 1, 1, 0
size3: .word 4


.text
.globl main
main:

         addi $s0, $zero, 1
Trial1:  la     $a0, test1arr1
         la     $a1, test1arr2
         lw     $a2, size1
         j Test

Trial2:  la     $a0, test2arr1
         la     $a1, test2arr2
         lw     $a2, size2
         j Test

Trial3:  la     $a0, test3arr1
         la     $a1, test3arr2
         lw     $a2, size3
         j Test

Test:   addi $s0, $s0, 1        #increment the test counter
        jal     func            # Save current PC in $ra, and jump to func
```

```
Test:   addi $s0, $s0, 1        #increment the test counter
        jal     func            # Save current PC in $ra, and jump to func

        la $a0, prompt
        li $v0, 4
        syscall

        addi $a0, $v1, 0
        li  $v0, 1
        syscall         #Output sum of absolute differences of the test

        beq $s0, 2, Trial2
        beq $s0, 3, Trial3
        li  $v0, 10
        syscall         #Exit


func:   #Compute sum of absolute differences
        addu $v1, $zero, $zero   #Set sum to 0
        addu $t0, $zero, $zero   #Set i to 0
        addu $t1, $zero, $zero   #Set diff to 0
        addiu $t4, $zero, 8      #Set byte overflow length
Loop:   addu $t2, $a0, $t0       #Set $t2 to the address of $a0[$t0]
        lb $t2, 0($t2)          #Load the value of array 1[i]
        addu.qb $t3, $a1, $t0       #Set $t3 to the address of $a1[$t0]
        lb $t3, 0($t3)          #Load the value of array 2[i]
        subu.qb $t1, $t2, $t3       #Set $t1 fo the difference between $t2 and $t3
        absq_s.qb $t1, $t1
Pos:    addu $v1, $v1, $t1       #Add the difference to the sum of differences
        addiu $t0, $t0, 1        #Increment $t0 by 1
        beq $t0, $t4, Exit      #If i = 8, more than a byte long exit
        bne $t0, $a2, Loop      #If i != array len Loop
Exit:   jr $ra                  #Jump to $ra
```

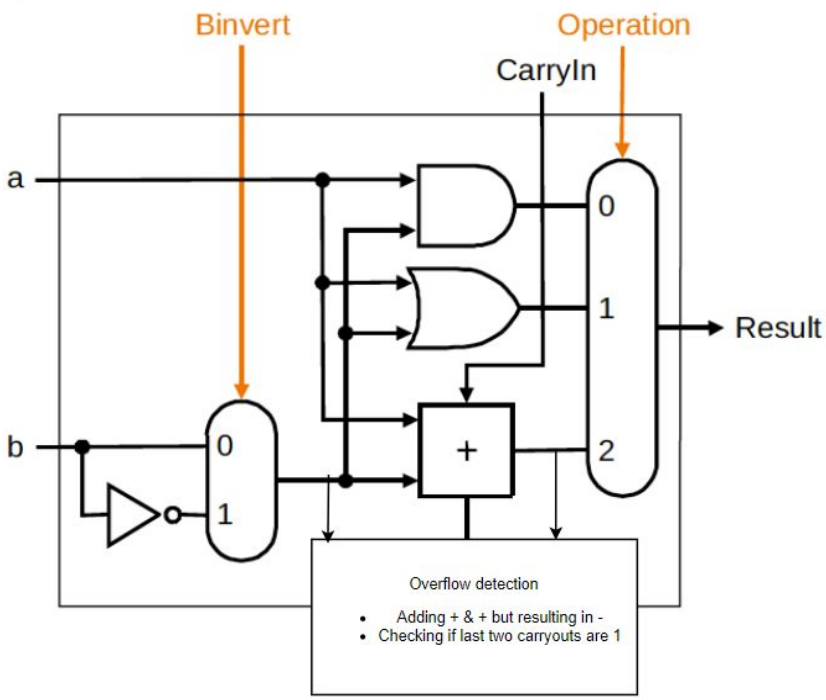
d. Instruction Count

   i. Standard: 278 Instructions

   ii. Quad Byte: 255 Instructions

3. Computer Arithmetic

a. Overflow



b. Logic Functions I

i. f = x2x1~x0 + x2~x1x0 + ~x2x1x0

ii. f = x2~x1~x0 + ~x2~x1x0 + ~x2x1~x0

iii. f = ~x2~x1~x0 + ~x2~x1x0 + ~x2x1~x0 + ~x2x1x0 (f = ~x2)

iv. f = x2~x1~x0 + x2~x1x0 + x2x1~x0 + x2x1x0 (f = x2)

c. Logic Functions II

i. f = (~x2y2) + ~(x2 XOR y2)(x1y1) + ~(x2 XOR y2)~(x1 XOR y1)(~x0y0)

ii. f = (x2~y2) + ~(x2 XOR y2)~x2(~x1y1 + ~(x1 XOR y1)~x0y0) + ~(x2 XOR y2)x2(x1~y1 + ~(x1 XOR y1)x0~y0)

iii. f = ~(x2 XOR y2)~(x1 XOR y1)~(x0 XOR y0)

d. 3 Bit Counter