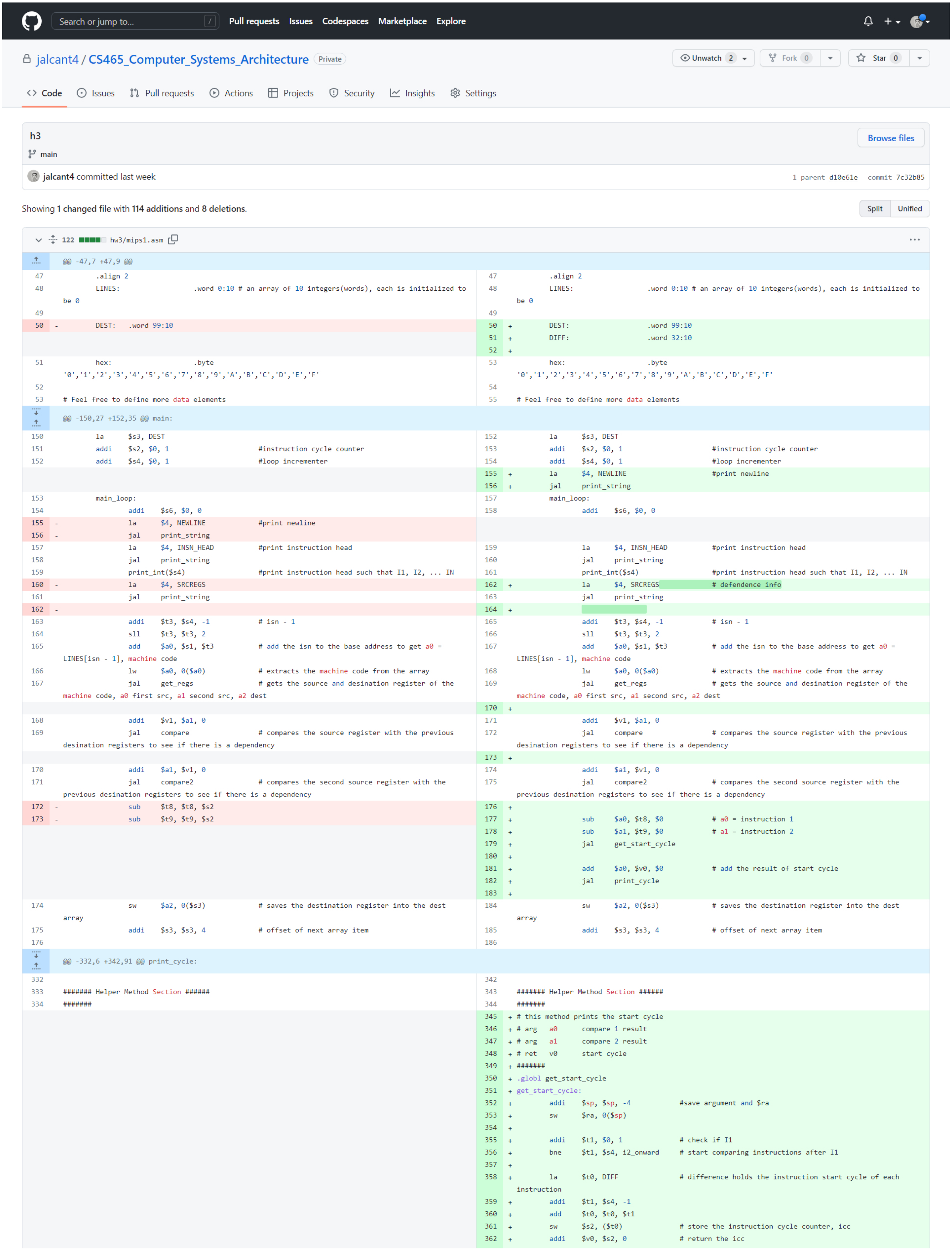


jalcant4 / CS465_Computer_Systems_Architecture Private

h3

main

jalcant4 committed last week

1 parent d10e61e commit 7c32b85

Showing **1 changed file** with **114 additions** and **8 deletions**.

122 hw3/mips1.asm

@@ -47,7 +47,9 @@

```
        .align 2
        LINES:                  .word 0:10 # an array of 10 integers(words), each is initialized to be 0

-       DEST:   .word 99:10

        hex:                    .byte '0','1','2','3','4','5','6','7','8','9','A','B','C','D','E','F'

# Feel free to define more data elements
```

```
        .align 2
        LINES:                  .word 0:10 # an array of 10 integers(words), each is initialized to be 0

+       DEST:                   .word 99:10
+       DIFF:                   .word 32:10
+
        hex:                    .byte '0','1','2','3','4','5','6','7','8','9','A','B','C','D','E','F'

# Feel free to define more data elements
```

@@ -150,27 +152,35 @@ main:

```
        la      $s3, DEST
        addi    $s2, $0, 1                      #instruction cycle counter
        addi    $s4, $0, 1                      #loop incrementer
        main_loop:
                addi    $s6, $0, 0
-               la      $4, NEWLINE             #print newline
-               jal     print_string
                la      $4, INSN_HEAD           #print instruction head
                jal     print_string
                print_int($s4)                  #print instruction head such that I1, I2, ... IN
-               la      $4, SRCREGS
                jal     print_string
-
                addi    $t3, $s4, -1            # isn - 1
                sll     $t3, $t3, 2
                add     $a0, $s1, $t3           # add the isn to the base address to get a0 = LINES[isn - 1], machine code
                lw      $a0, 0($a0)             # extracts the machine code from the array
                jal     get_regs                # gets the source and desination register of the machine code, a0 first src, a1 second src, a2 dest
                addi    $v1, $a1, 0
                jal     compare                 # compares the source register with the previous desination registers to see if there is a dependency
                addi    $a1, $v1, 0
                jal     compare2                # compares the second source register with the previous desination registers to see if there is a dependency
-               sub     $t8, $t8, $s2
-               sub     $t9, $t9, $s2
                sw      $a2, 0($s3)             # saves the destination register into the dest array
                addi    $s3, $s3, 4             # offset of next array item

```

```
        la      $s3, DEST
        addi    $s2, $0, 1                      #instruction cycle counter
        addi    $s4, $0, 1                      #loop incrementer
+       la      $4, NEWLINE                     #print newline
+       jal     print_string
        main_loop:
                addi    $s6, $0, 0
                la      $4, INSN_HEAD           #print instruction head
                jal     print_string
                print_int($s4)                  #print instruction head such that I1, I2, ... IN
+               la      $4, SRCREGS             # defendence info
                jal     print_string
+
                addi    $t3, $s4, -1            # isn - 1
                sll     $t3, $t3, 2
                add     $a0, $s1, $t3           # add the isn to the base address to get a0 = LINES[isn - 1], machine code
                lw      $a0, 0($a0)             # extracts the machine code from the array
                jal     get_regs                # gets the source and desination register of the machine code, a0 first src, a1 second src, a2 dest
+
                addi    $v1, $a1, 0
                jal     compare                 # compares the source register with the previous desination registers to see if there is a dependency
+
                addi    $a1, $v1, 0
                jal     compare2                # compares the second source register with the previous desination registers to see if there is a dependency
+
+               sub     $a0, $t8, $0            # a0 = instruction 1
+               sub     $a1, $t9, $0            # a1 = instruction 2
+               jal     get_start_cycle
+
+               add     $a0, $v0, $0            # add the result of start cycle
+               jal     print_cycle
+
                sw      $a2, 0($s3)             # saves the destination register into the dest array
                addi    $s3, $s3, 4             # offset of next array item

```

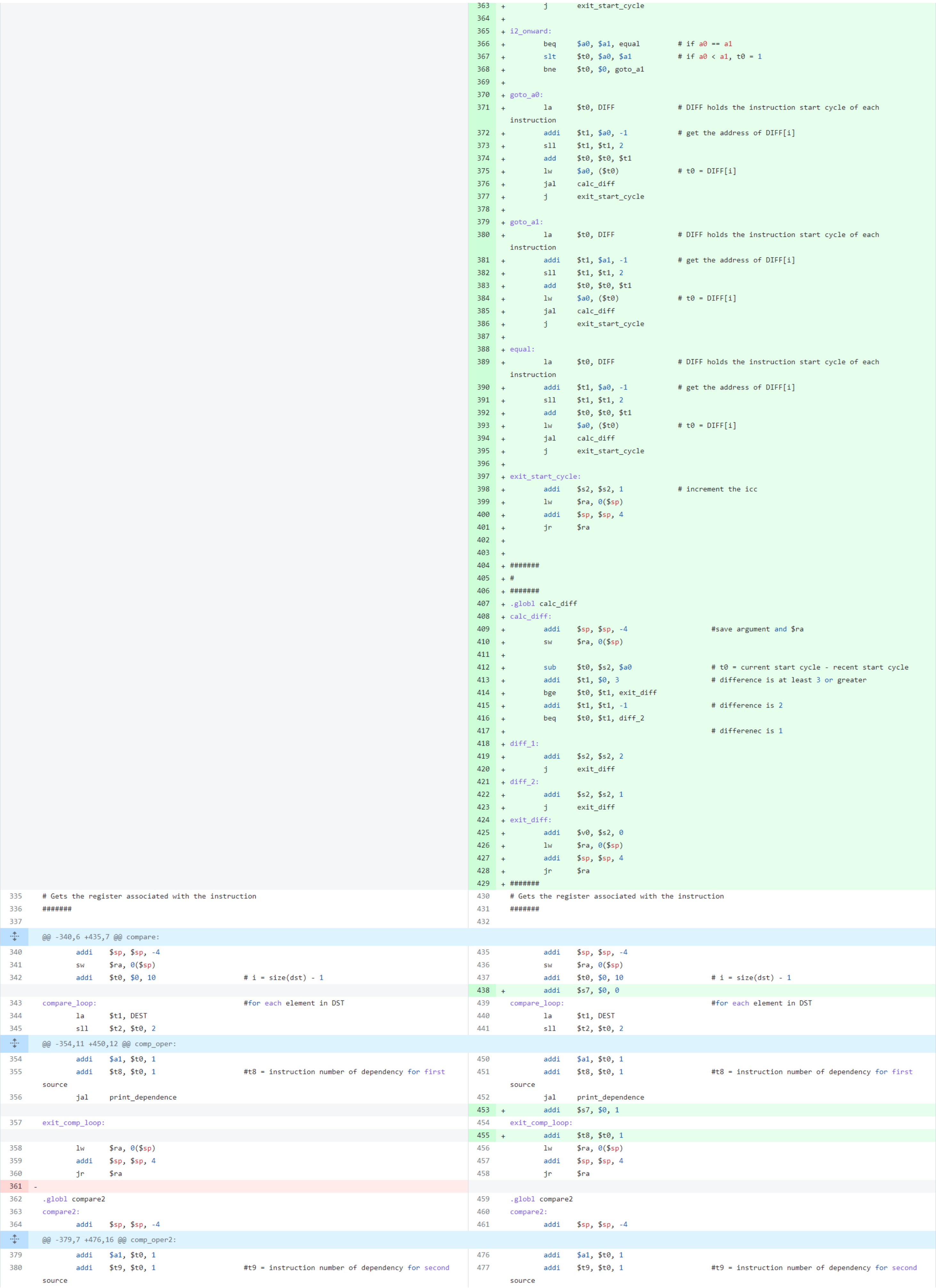
@@ -332,6 +342,91 @@ print_cycle:

```

###### Helper Method Section ######
#######
```

```

###### Helper Method Section ######
#######
+# this method prints the start cycle
+# arg   a0      compare 1 result
+# arg   a1      compare 2 result
+# ret   v0      start cycle
+#######
+.globl get_start_cycle
+get_start_cycle:
+       addi    $sp, $sp, -4            #save argument and $ra
+       sw      $ra, 0($sp)
+
+       addi    $t1, $0, 1              # check if I1
+       bne     $t1, $s4, i2_onward     # start comparing instructions after I1
+
+       la      $t0, DIFF               # difference holds the instruction start cycle of each instruction
+       addi    $t1, $s4, -1
+       add     $t0, $t0, $t1
+       sw      $s2, ($t0)              # store the instruction cycle counter, icc
+       addi    $v0, $s2, 0             # return the icc
```



```
+        j       exit_start_cycle
+
+i2_onward:
+        beq     $a0, $a1, equal         # if a0 == a1
+        slt     $t0, $a0, $a1           # if a0 < a1, t0 = 1
+        bne     $t0, $0, goto_a1
+
+goto_a0:
+        la      $t0, DIFF               # DIFF holds the instruction start cycle of each instruction
+        addi    $t1, $a0, -1            # get the address of DIFF[i]
+        sll     $t1, $t1, 2
+        add     $t0, $t0, $t1
+        lw      $a0, ($t0)              # t0 = DIFF[i]
+        jal     calc_diff
+        j       exit_start_cycle
+
+goto_a1:
+        la      $t0, DIFF               # DIFF holds the instruction start cycle of each instruction
+        addi    $t1, $a1, -1            # get the address of DIFF[i]
+        sll     $t1, $t1, 2
+        add     $t0, $t0, $t1
+        lw      $a0, ($t0)              # t0 = DIFF[i]
+        jal     calc_diff
+        j       exit_start_cycle
+
+equal:
+        la      $t0, DIFF               # DIFF holds the instruction start cycle of each instruction
+        addi    $t1, $a0, -1            # get the address of DIFF[i]
+        sll     $t1, $t1, 2
+        add     $t0, $t0, $t1
+        lw      $a0, ($t0)              # t0 = DIFF[i]
+        jal     calc_diff
+        j       exit_start_cycle
+
+exit_start_cycle:
+        addi    $s2, $s2, 1             # increment the icc
+        lw      $ra, 0($sp)
+        addi    $sp, $sp, 4
+        jr      $ra
+
+
+#######
+#
+#######
+.globl calc_diff
+calc_diff:
+        addi    $sp, $sp, -4                    #save argument and $ra
+        sw      $ra, 0($sp)
+
+        sub     $t0, $s2, $a0                   # t0 = current start cycle - recent start cycle
+        addi    $t1, $0, 3                      # difference is at least 3 or greater
+        bge     $t0, $t1, exit_diff
+        addi    $t1, $t1, -1                    # difference is 2
+        beq     $t0, $t1, diff_2
+                                                # differenec is 1
+diff_1:
+        addi    $s2, $s2, 2
+        j       exit_diff
+diff_2:
+        addi    $s2, $s2, 1
+        j       exit_diff
+exit_diff:
+        addi    $v0, $s2, 0
+        lw      $ra, 0($sp)
+        addi    $sp, $sp, 4
+        jr      $ra
+#######
 # Gets the register associated with the instruction
 #######

@@ -340,6 +435,7 @@ compare:
         addi    $sp, $sp, -4
         sw      $ra, 0($sp)
         addi    $t0, $0, 10                     # i = size(dst) - 1
+        addi    $s7, $0, 0
 compare_loop:                                   #for each element in DST
         la      $t1, DEST
         sll     $t2, $t0, 2
@@ -354,11 +450,12 @@ comp_oper:
         addi    $a1, $t0, 1
         addi    $t8, $t0, 1                     #t8 = instruction number of dependency for first source
         jal     print_dependence
+        addi    $s7, $0, 1
 exit_comp_loop:
+        addi    $t8, $t0, 1
         lw      $ra, 0($sp)
         addi    $sp, $sp, 4
         jr      $ra
-
 .globl compare2
 compare2:
         addi    $sp, $sp, -4
@@ -379,7 +476,16 @@ comp_oper2:
         addi    $a1, $t0, 1
         addi    $t9, $t0, 1                     #t9 = instruction number of dependency for second source
```

```
        jal     print_dependence
+       addi    $s7, $0, 1
exit_comp_loop2:
+       beq     $s7, $0, print_none
+       lw      $ra, 0($sp)
+       addi    $sp, $sp, 4
+       jr      $ra
+print_none:
+       la      $4, MSG_NONE
+       jal     print_string
+       addi    $t9, $t0, 1
        lw      $ra, 0($sp)
        addi    $sp, $sp, 4
        jr      $ra
```

0 comments on commit `7c32b85`

Lock conversation

Write Preview

Leave a comment

Attach files by dragging & dropping, selecting or pasting them.

Comment on this commit

Unsubscribe You're receiving notifications because you're watching this repository.