VE370 HW1 周维骁 518021911039

1. $\text{CPU Time}_A = \frac{\text{Instruction Count}_A \times \text{CPI}_A}{\text{Clock Rate}_A} = \frac{100000 \times 1.5}{500 \times 10^6}$ s

$\text{CPU Time}_B = \frac{\text{Instruction Count}_B \times \text{CPI}_B}{\text{Clock Rate}_B} = \frac{\text{Instruction Count}_B \times 2.3}{850 \times 10^6} = \frac{100000 \times 1.5}{500 \times 10^6}$ s

$\text{Instruction Count}_B \approx 110870$

2. Clock Cycles = $1100 \times 1 + 200 \times 7 + 150 \times 7 + 130 \times 3 = 3910$

= Instruction Count $\times$ CPI = $1570 \times$ CPI

Then, we have CPI ≈ 2.5

$\text{CPU Time} = \frac{\text{Clock Cycles}}{\text{Clock Rate}} = \frac{3910}{2 \times 10^9} = 1.955 \times 10^{-6}$ s

3. 1 processor: $\text{CPU Time}_1 = \frac{\text{Clock Cycles}_1}{\text{Clock Rate}} = \frac{256 \times 1 + 128 \times 4 + 56 \times 2}{3 \times 10^9} = 2.73 \times 10^{-6}$ s

2 processors: $\text{CPU Time}_2 = \frac{\text{Clock Cycles}_2}{\text{Clock Rate}} = \frac{128 \times 1 + 64 \times 6 + 128 \times 2}{3 \times 10^9} = 1.79 \times 10^{-6}$ s

4 processors: $\text{CPU Time}_4 = \frac{\text{Clock Cycles}_4}{\text{Clock Rate}} = \frac{64 \times 1 + 32 \times 8 + 64 \times 2}{3 \times 10^9} = 1.11 \times 10^{-6}$ s

8 processors: $\text{CPU Time}_8 = \frac{\text{Clock Cycles}_8}{\text{Clock Rate}} = \frac{32 \times 1 + 16 \times 10 + 32 \times 2}{3 \times 10^9} = 6.61 \times 10^{-7}$ s

4.
```
sub $t0, $s0, $s1
add $t1, $t0, $s2
addi $s0, $t1, -72
```

5.
```
sll $t2, $t1, 2
add $t2, $t2, $s5
lw $t3, 0($t2)
add $t3, $t0, $t3
sw $t3, 32($s6)
```

6. Base address of A is in $s6, a is in $s0.

Then the C code is:

A[2] = &A[0]

a = A[1] + A[2];

The final value is 0x0000F1a4 + 0x00000100 = 0x0000F2a4

7. addi $t0, $0, 1

sll $t0, $t0, 28

lb $s2, 2($t0)

The content should be 0x00000066

8. $t0 < $t1

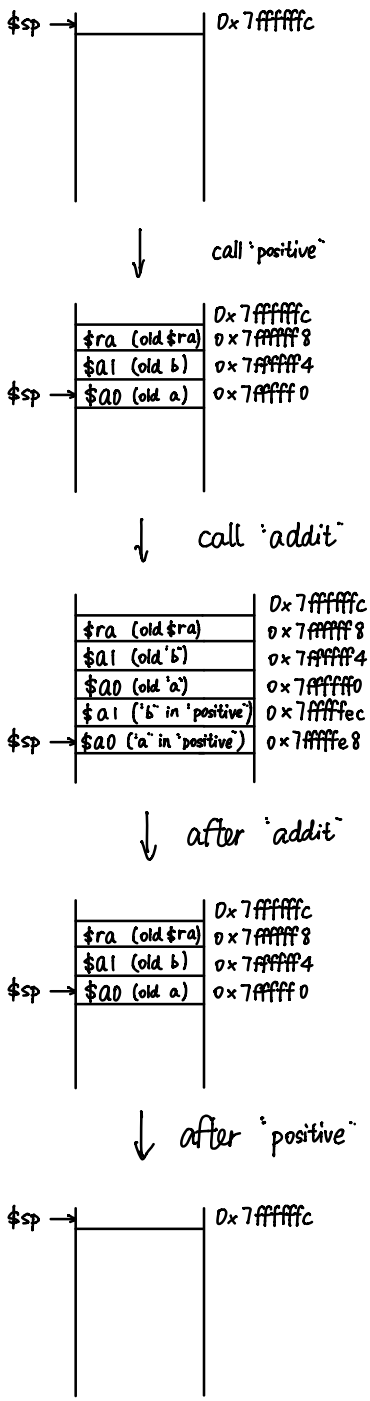
$t2 = 1

Therefore, the value of $t2 is 1.

9. Suppose a is in $a0, b in $a1, the result of "positive" is in $v0, the result of "addit" is $v1

positive:

```
    addi $sp, $sp, -12   # create space on stack
    sw $ra, 8($sp)       # store data on stack
    sw $a1, 4($sp)
    sw $a0, 0($sp)
    jal addit            # Jump to "addit"
    slt $v0, $0, $v1     # If 0<$v1, $v0=1, else $v0=0
    lw $a0, 0($sp)       # restore data from stack
    lw $a1, 4($sp)
    lw $ra, 8($sp)
    addi $sp, $sp, 12    # destroy spaces on stack
    jr $ra               # return from function
addit:
    addi $sp, $sp, -8    # create space on stack
    sw $a1, 4($sp)       # store data on stack
    sw $a0, 0($sp)
    add $v1, $a0, $a1    # get the result of a+b
    lw $a0, 0($sp)       # restore data from stack
    lw $a1, 4($sp)
    addi $sp, $sp, 8     # destroy spaces on stack
    jr $ra               # return from function
```


$sp
0x7ffffffc
call "positive"
0x7ffffffc
$ra (old $ra)
0x7ffffff8
$a1 (old b)
0x7ffffff4
$sp
$a0 (old a)
0x7fffff0
call "addit"
0x7ffffffc
$ra (old $ra)
0x7ffffff8
$a1 (old "b")
0x7ffffff4
$a0 (old "a")
0x7fffffo
$a1 ("b" in "positive")
0x7fffffec
$sp
$a0 ("a" in "positive")
0x7fffffe8
after "addit"
0x7ffffffc
$ra (old $ra)
0x7ffffff8
$a1 (old b)
0x7ffffff4
$sp
$a0 (old a)
0x7fffff0
after "positive"
$sp
0x7ffffffc