計算機結構 Exercise 02

B03902062 資工三 董文捷

2.27

	`add`	$t0,	$zero,	$zero	# i = 0
L1	`add`	$t1,	$zero,	$zero	# j = 0 (restart 2nd for loop)
L2	`sll`	$t2,	$t1,	4	# t2 = (4 * j) * 4 (byte offset of [4 * j])
	`add`	$t3,	$s2,	$t2	# t3 = v + (4 * j) * 4 (byte address of D[4 * j])
	`add`	$t4,	$t0,	$t1	# t4 = i + j
	`sw`	$t4,	0($t3)		# D[4 * j] = i + j
	`addi`	$t1,	$t1,	1	# j = j + 1
	`slt`	$t5,	$t1,	$s1	# if j < b
	`bne`	$t5,	$zero,	L2	# branch to L2
	`addi`	$t0,	$t0,	1	# i = i + 1
	`slt`	$t5,	$t0,	$s0	# if i < a
	`bne`	$t5,	$zero,	L1	# branch to L1

2.34 For function `f`, assume arguments a, b, c, d in $a0, $a1, $a2, $a3 and result in $v0. For function `func`, assume arguments a, b in $a0, $a1 and result in $v0.

f	`addi`	$sp,	$sp,	-12	# adjust stack for 3 items
	`sw`	$ra,	8($sp)		# save return address
	`sw`	$a1,	4($sp)		# save argument 1
	`sw`	$a0,	0($sp)		# save argument 0
	`jal`	func			# v0 = func(a, b)
	`add`	$a0,	$v0,	$zero	# a0 = func(a, b)
	`add`	$a1,	$a2,	$a3	# a1 = c + d
	`jal`	func			# v0 = func(func(a, b), c + d)
	`lw`	$a0,	0($sp)		# restore original argument 0
	`lw`	$a1,	4($sp)		# restore original argument 1
	`lw`	$ra,	8($sp)		# restore original return address
	`addi`	$sp,	$sp,	12	# pop 3 items from stack
	`jr`	$ra			# return

2.46.1 Original clock cycles for the program
$= 500 \times 10^6 \times 1 + 300 \times 10^6 \times 10 + 100 \times 10^6 \times 3 = 3800 \times 10^6$
Total execution time $= 3800 \times 10^6 \times$ original clock cycle time

New clock cycles for the program
$= 500 \times 10^6 \times 75\% \times 1 + 300 \times 10^6 \times 10 + 100 \times 10^6 \times 3 = 3675 \times 10^6$
Total execution time $= 3675 \times 10^6 \times$ (original clock cycle time $\times 125\%$)
$= 4593.75 \times 10^6 \times$ original clock cycle time

This is not a good design choice because execution time for arithmetic instructions is not the main part of total execution time. Although new design can make arithmetic part faster, increment on clock cycle time will cause other part become slower and thus result in a worse overall performance.

2.46.2

1. New clock cycles for the program
   $= (500 \times 10^6 \times 1) \times \frac{1}{2} + 300 \times 10^6 \times 10 + 100 \times 10^6 \times 3 = 3550 \times 10^6$
   Total execution time $= 3550 \times 10^6 \times$ original clock cycle time
   $\frac{3800 \times 10^6}{3550 \times 10^6} = 1.0704$, overall speedup $= 7.04\%$
2. New clock cycles for the program
   $= (500 \times 10^6 \times 1) \times \frac{1}{10} + 300 \times 10^6 \times 10 + 100 \times 10^6 \times 3 = 3350 \times 10^6$
   Total execution time $= 3350 \times 10^6 \times$ original clock cycle time
   $\frac{3800 \times 10^6}{3350 \times 10^6} = 1.1343$, overall speedup $= 13.43\%$

3.14 Hardware : Add multiplicand to product, then shift the multiplicand and the multiplier simultaneously $\rightarrow$ 8 time units per repetition
Total time = 8 time units $\times$ 8 repetitions = 64 time units.
Software : Add multiplicand to product, then shift the multiplicand, finally shift the multiplier $\rightarrow$ 12 time units per repetition
Total time = 12 time units $\times$ 8 repetitions = 96 time units.

3.27 $-1.5625 \times 10^{-1} = (-1)^1 \times 1.01_2 \times 2^{-3}$
S = `1`
Fraction = `0100000000`
Exponent = -3 + Bias = 13 = `01101`
Therefore, -1.5625×10^{-1} can be represented as `1011010100000000`.

half-precision range :
Exponent `00000` and `11111` reserved

Smallest value

Exponent : `00001` $\rightarrow$ actual exponent $= 1 - 15 = -14$

Fraction : `000...00` $\rightarrow$ significand $= 1.0$

$\pm 1.0 \times 2^{-14} \approx \pm 6.1 \times 10^{-5}$

Largest value

Exponent : `11110` $\rightarrow$ actual exponent $= 30 - 15 = +15$

Fraction : `111...11` $\rightarrow$ significand ≈ 2.0

$\pm 2.0 \times 2^{+15} \approx \pm 6.6 \times 10^{+4}$

single-precision range :

Exponent `00000000` and `11111111` reserved

Smallest value

Exponent : `00000001` $\rightarrow$ actual exponent $= 1 - 127 = -126$

Fraction : `000...00` $\rightarrow$ significand $= 1.0$

$\pm 1.0 \times 2^{-126} \approx \pm 1.2 \times 10^{-38}$

Largest value

Exponent : `11111110` $\rightarrow$ actual exponent $= 254 - 127 = +127$

Fraction : `111...11` $\rightarrow$ significand ≈ 2.0

$\pm 2.0 \times 2^{+127} \approx \pm 3.4 \times 10^{+38}$

half-precision accuracy :

$\approx 2^{-10}$, equivalent to $10 \times \log_{10} 2 \approx 3$ decimal digits of precision

single-precision accuracy :

$\approx 2^{-23}$, equivalent to $23 \times \log_{10} 2 \approx 7$ decimal digits of precision