

Project 2 Report

Anna Phan

1.0: Program Input/Output

Program 1: The input of the program is the user putting in two positive numbers. The output would be the number calculate using multiplication procedures.

Program 2: The input of the program is the user entering in three signed numbers. The output of would be the answer of the equation on the worksheet along with a message saying whether or not here is overflow.

Program 3: The input of the program is the user entering in Instruction Count, CPI (Clock Per Instruction), and Clock Rate in GHz for processor A and the same for processor B. The output be telling the user which one is faster.

2.0: Program Design

Program 1: The program will prompt the use to enter in two positive integers. The program will take the put the first number into a loop. The loop will start at 1 and go till the it is equal to the second number. Within the loop, the first number will add to itself and move the loop to the next number. Once the loop is equal to the second number, the loop will end and print out the number.

Program 2: The program will prompt the user to enter in three signed numbers. It will have a loop so the result of the loop will be the second powered by the third number. Then it will as the first number to the result. It will square the first number and have the third number subtracted from it. The program will take those number and diver the first result by the second result. Next the program will multiple the second number with 3 and have that added to the result we already have. The program will then check for overflow.

Program 3: The program will prompt the user for Instruction Count, CPI, and Clock Rate in GHz for processor A. It will then calculate the CPU time for processor A. The program will prompt the user for the same information but from processor B. It will calculate the CPU time for processor B. The program will compare the two to see which has the bigger CPU time and print out the appropriate one that is fast and how many times it faster than the other.

3.0: Symbol Table

Register	Purpose & Label
\$v0	1: prints an integer 2: prints a float 4: prints a string 5: saves an integer from user 6: saves a float from user
\$a0	Argument of syscall to print string
\$t0	C1: temporary place for the add number C2: temporary place for the second number powered by the third number. temporary place for that number added with the first number
\$t1	C2: temporary place for the square root of the first number. Temporary place for the result minus by the third number.
\$t2	C2: temporary place for the second number multiplied by 3
\$s0	C1: save the first number the user entered C2: save the first number the user entered
\$s1	C1: save the second number the user entered C2: save the second number the user entered
\$s2	C1: save the number that the loop starts on C2: save the third number the user entered

\$s4	C2: save the result after the calculations
\$f0	C3: used to transfer the user input into a different register
\$f1	C3: temporary place for processor A's Instruction Count
\$f2	C3: temporary place for processor A's Clock Per Instruction
\$f3	C3: temporary place for processor A's Clock Rate in GHz
\$f4	C3: temporary zero
\$f5	C3: temporary place for processor A's CPU Time
\$f6	C3: temporary place for processor B's Instruction Count
\$f7	C3: temporary place for processor B's Clock Per Instruction
\$f8	C3: temporary place for processor B's Clock Rate in GHz
\$f9	C3: temporary place for processor B's CPU Time
\$f12	C3: passes the number so it can be printed

4.0: Learning Coverage

1. CPU Performance
2. Storing multiple floats from user input
3. Printing floats
4. Checking for overflow
5. Using a loop to perform multiplication procedures

5.0: Test Results

Program 1:

The screenshot shows the MARS MIPS simulator interface. The main window displays assembly code with columns for Address, Code, Basic, and Source. The Data Segment window shows memory addresses and values. The Registers window shows the state of various registers. The MARS Messages window shows the program's execution status.

Program 2:

File Edit Run Settings Tools Help

Run speed at max (no interaction)

Edit Execute

Text Segment

Bkpt	Address	Code	Basic	Source
	0x00400000	0x24020004	addiu \$2,\$0,0x00000004	7: li \$v0, 4
	0x00400004	0x3c011001	lui \$1,0x00001001	8: la \$a0, intro
	0x00400008	0x34240000	ori \$4,\$1,0x00000000	
	0x0040000c	0x0000000c	syscall	9: syscall
	0x00400010	0x24020005	addiu \$2,\$0,0x00000005	10: li \$v0, 5
	0x00400014	0x0000000c	syscall	11: syscall
	0x00400018	0x00028021	addu \$16,\$0,\$2	12: move \$s0, \$v0 # a
	0x0040001c	0x24020005	addiu \$2,\$0,0x00000005	13: li \$v0, 5
	0x00400020	0x0000000c	syscall	14: syscall
	0x00400024	0x00028021	addu \$17,\$0,\$2	15: move \$s1, \$v0 # b
	0x00400028	0x24020005	addiu \$2,\$0,0x00000005	16: li \$v0, 5
	0x0040002c	0x0000000c	syscall	17: syscall
	0x00400030	0x0000000c	syscall	18: syscall

Data Segment

Address	Value (+0)	Value (+4)	Value (+8)	Value (+c)	Value (+10)	Value (+14)	Value (+18)	Value (+1c)
0x10010000	0x65746e45	0x68742072	0x20656572	0x6e676973	0x69206465	0x6765746e	0x3a737265	0x44000a20
0x10010020	0x73697f69	0x7a207962	0x206e6f69	0x7a207962	0x066f7265	0x656540a	0x9206572	0x766f2073
0x10010040	0x0a00776f	0x72656584	0x73692065	0x206f6e20	0x7265776f	0x776f6c66	0x00000000	0x00000000
0x10010060	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x10010080	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x100100a0	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x100100c0	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x100100e0	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x10010100	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x10010120	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000

Registers

Name	Number	Value
\$zero	0	0x00000000
\$at	1	0x10010000
\$v0	2	0x0000000a
\$v1	3	0x00000000
\$a0	4	0x10010043
\$a1	5	0x00000000
\$a2	6	0x00000000
\$a3	7	0x00000000
\$t0	8	0x00000010
\$t1	9	0x00000000
\$t2	10	0xffffffff
\$t3	11	0x00000000
\$t4	12	0x00000000
\$t5	13	0x00000000
\$t6	14	0x00000000
\$t7	15	0x00000000
\$s0	16	0x00000000
\$s1	17	0xffffffff
\$s2	18	0x00000002
\$s3	19	0x00000003
\$s4	20	0xffffffff
\$s5	21	0x00000000
\$s6	22	0x00000000
\$s7	23	0x00000000
\$s8	24	0x00000000
\$t9	25	0x00000000
\$k0	26	0x00000000
\$k1	27	0x00000000
\$gp	28	0x10000000
\$sp	29	0x7fffffc0
\$fp	30	0x00000000
\$ra	31	0x00000000
pc		0x00400030
hi		0xffffffff
lo		0xffffffff

Mars Messages Run I/O

Enter three signed integers:
0
-4
2
-4
Clear
There is no overflow
— program is finished running —

Program 3:

File Edit Run Settings Tools Help

Run speed at max (no interaction)

Edit Execute

Text Segment

Bkpt	Address	Code	Basic	Source
	0x00400000	0x3c011001	lui \$1,0x00001001	14: lwcl \$f4, zero
	0x00400004	0xc4240000	lwcl \$f4, 0x00000000...	
	0x00400008	0x24020004	addiu \$2,\$0,0x00000004	16: li \$v0, 4
	0x0040000c	0x3c011001	lui \$1,0x00001001	17: la \$a0, processor1
	0x00400010	0x34240000	ori \$4,\$1,0x00000000	
	0x00400014	0x0000000c	syscall	18: syscall
	0x00400018	0x24020004	addiu \$2,\$0,0x00000004	20: li \$v0, 4
	0x0040001c	0x3c011001	lui \$1,0x00001001	21: la \$a0, ICMessag
	0x00400020	0x34240000	ori \$4,\$1,0x00000000	
	0x00400024	0x0000000c	syscall	22: syscall
	0x00400028	0x24020004	addiu \$2,\$0,0x00000004	23: li \$v0, 6
	0x0040002c	0x0000000c	syscall	24: syscall
	0x00400030	0x60000000	add.s \$f1,\$f0,\$f4	25: add.s \$f1, \$f0, \$f4
	0x00400034	0x24020004	addiu \$2,\$0,0x00000004	27: li \$v0, 4
	0x00400038	0x3c011001	lui \$1,0x00001001	28: la \$a0, CPMessag
	0x0040003c	0x34240000	ori \$4,\$1,0x00000000	

Data Segment

Address	Value (+0)	Value (+4)	Value (+8)	Value (+c)	Value (+10)	Value (+14)	Value (+18)	Value (+1c)
0x10010000	0x636f7273	0x6f727365	0x6f727365	0x500a000a	0x56636f72	0x736f7273	0x0a3a6220	
0x10010020	0x4927265	0x7274736e	0x69746375	0x432066f	0x746e756f	0x4500203a	0x7265746e	
0x10010040	0x6c432820	0x2060636f	0x20726550	0x74736e49	0x7265506f	0x296e6f69	0x4500203a	
0x10010060	0x43206320	0x40d36f6c	0x74615220	0x6e6f2065	0x74615220	0x0a00203a	0x736f7273	
0x10010080	0x73656669	0x20736320	0x74736320	0x20736320	0x0a002042	0x736f7273	0x74206020	
0x100100a0	0x20736320	0x74736320	0x20736320	0x00002e41	0x00000000	0x00000000	0x00000000	
0x100100c0	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	
0x100100e0	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	
0x10010100	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	
0x10010120	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	
0x10010140	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	
0x10010160	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	
0x10010180	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	

Registers

Name	Number	Value
\$zero	0	0x00000000
\$at	1	0x10010000
\$v0	2	0x0000000a
\$v1	3	0x00000000
\$a0	4	0x1001005a
\$a1	5	0x00000000
\$a2	6	0x00000000
\$a3	7	0x00000000
\$t0	8	0x00000000
\$t1	9	0x00000000
\$t2	10	0x00000000
\$t3	11	0x00000000
\$t4	12	0x00000000
\$t5	13	0x00000000
\$t6	14	0x00000000
\$t7	15	0x00000000
\$s0	16	0x00000000
\$s1	17	0x00000000
\$s2	18	0x00000000
\$s3	19	0x00000000
\$s4	20	0x00000000
\$s5	21	0x00000000
\$s6	22	0x00000000
\$s7	23	0x00000000
\$s8	24	0x00000000
\$t9	25	0x00000000
\$k0	26	0x00000000
\$k1	27	0x00000000
\$gp	28	0x10000000
\$sp	29	0x7fffffc0
\$fp	30	0x00000000
\$ra	31	0x00000000
pc		0x00400030
hi		0x00000000
lo		0x00000000

Mars Messages Run I/O

Processor A:
Enter Instruction Count: 100
Enter CPI (Clock Per Instruction): 4
Enter a Clock Rate in GHz: 4

Processor B:
Enter Instruction Count: 100
Enter CPI (Clock Per Instruction): 2
Enter a Clock Rate in GHz: 4

B is 2.0 times as fast as A.
— program is finished running —