

Thursday, September 19, 2019 8:43 PM

Project 1 Report Questions

- Which level does your program achieve? How many hours have you spend in total on this project? What was the most difficult (or time-consuming) part of this project?

The program achieves (A) + (B) (i), (ii). In total, I spent about 5 hours on this project. The most difficult part of the project was forming the pseudocode that fit the required task the program was to do, since it involved creative thinking in addition to prior knowledge of how information is handled within the simulated MIPS processor.

- What is the total (dynamic) instruction count of your program? If extra credit will be given to achieve low dynamical instruction count, how would you optimize your program?

There were 4,994 instructions run during the execution of this program. In the future, I would attempt to reduce the number of instructions required to move and compare data, such as in the pattern matching section with multiple shifts and comparison branches.

- Overall, do you think this multiply-fold is a good Hash Function? Provide some reasons or analysis based on data evidences.

This function does not appear to be a good function for generating data in this manner. There appears to be a lot of granular information lost in the folding process, and the resulting hash table also appears to have frequent collisions due to the small size of data elements.

Edit Run Settings Tools Help

Run speed 30 inst/sec

Edit Execute

Text Segment

Bkpt	Address	Code	Basic	Source
0x000000c8	0x2015001f	addi \$t1,\$0,0x0000001f	l11: addi \$t1, \$0, 31	
0x000000cc	0x00004e20	addi \$t0,\$0,40	l13: addi \$t0, \$0, 40	
0x000000d0	0x04500001	lw \$t0,0x00000000(\$t0)	l18: lw \$t0, 0(\$t0)	
0x000000d4	0x325e001f	andi \$t2,\$t0,0x0000...	l20: andi \$t2, \$t0, 31	
0x000000d8	0x1245000a	bqz \$t2,\$t1,0x0000000a	l22: bqz \$t2, \$t1, match_l1111	
0x000000dc	0x0012b042	srl \$t2,\$t0,0x00000001	l24: srl \$t2, \$t0, 1	
0x000000e0	0x325e001f	andi \$t2,\$t0,0x0000...	l25: andi \$t2, \$t0, 31	
0x000000e4	0x12450007	bqz \$t2,\$t1,0x00000007	l26: bqz \$t2, \$t1, match_l1111	
0x000000e8	0x0012b082	srl \$t2,\$t0,0x00000002	l28: srl \$t2, \$t0, 2	
0x000000ec	0x325e001f	andi \$t2,\$t0,0x0000...	l29: andi \$t2, \$t0, 31	
0x000000f0	0x12450004	bqz \$t2,\$t1,0x00000004	l30: bqz \$t2, \$t1, match_l1111	
0x000000f4	0x0012b0c2	srl \$t2,\$t0,0x00000003	l32: srl \$t2, \$t0, 3	
0x000000f8	0x325e001f	andi \$t2,\$t0,0x0000...	l33: andi \$t2, \$t0, 31	
0x000000fc	0x12450001	bqz \$t2,\$t1,0x00000001	l34: bqz \$t2, \$t1, match_l1111	
0x00000100	0x10000001	bqz \$t0,\$t0,0x00000001	l37: bqz \$t0, \$t0, endofloop	
0x00000104	0x21590001	addi \$t5,\$t0,0x00000001	l40: addi \$t5, \$t0, 1	
0x00000108	0x214a0004	addi \$t0,\$t0,0x0000...	l44: addi \$t0, \$t0, 4	
0x0000010c	0x1557ff0a	jne \$t0,\$t3,0xffff0a	l45: jne \$t0, \$t3, loop_l1111	
0x00000110	0x340a2008	ori \$t0,\$t0,0x00002008	l48: ori \$t0, \$t0, 0x2008	
0x00000114	0xad490000	sw \$t5,0x00000000(\$t0)	l49: sw \$t5, 0(\$t0)	

Data Segment

Address	Value (+0)	Value (+4)	Value (+8)	Value (+c)	Value (+10)	Value (+14)	Value (+18)	Value (+1c)
0x00002000	0x000020a4	0x000000ff	0x00000003	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x00002020	0x000000e5	0x0000000f	0x0000007c	0x00000019	0x00000024	0x0000005e	0x00000070	0x00000005
0x00002040	0x0000008d	0x0000007e	0x000000fe	0x000000a0	0x00000035	0x0000002c	0x0000007f	0x00000008
0x00002060	0x0000000e	0x000000a0	0x0000004b	0x0000004e	0x00000090	0x000000a0	0x00000052	0x0000005d
0x00002080	0x000000d4	0x000000e6	0x00000006	0x00000007	0x00000037	0x00000008	0x00000004	0x00000006
0x000020a0	0x0000005d	0x000000ff	0x000000ec	0x00000013	0x00000070	0x000000c3	0x000000e2	0x0000007a
0x000020c0	0x000000cf	0x00000019	0x000000ed	0x000000e8	0x00000040	0x0000000a	0x0000000e	0x00000023
0x000020e0	0x00000001	0x000000df	0x0000002e	0x000000e5	0x00000075	0x00000009	0x00000014	0x000000be
0x00002100	0x00000089	0x000000ac	0x0000000b	0x00000048	0x000000a1	0x000000c2	0x00000012	0x000000eb
0x00002120	0x00000091	0x0000000a	0x00000001	0x00000005	0x0000000a	0x00000009	0x00000005	0x0000005d
0x00002140	0x000000fe	0x0000000f	0x00000070	0x00000000	0x00000000	0x00000031	0x00000075	0x00000014
0x00002160	0x000000fa	0x000000ef	0x000000e9	0x00000043	0x00000004	0x00000004	0x00000032	0x00000000
0x00002180	0x00000062	0x000000ba	0x00000000	0x00000006	0x0000004f	0x00000061	0x000000da	0x000000f1
0x000021a0	0x000000e7	0x00000073	0x00000071	0x0000002a	0x00000000	0x00000000	0x00000000	0x00000000

0x00002000 (.data) Hexadecimal Addresses Hexadecimal Values ASCII

Mars Messages Run IO

Clear

```
# Author: Sheng Chen
# For: UIC ECE 366 Fall 2019 Project 1
# "Multiply and Fold" Hash Function
# Calculate C = H(A, B) for a given B = 0xFA19E366,
# for all the A values ranging from 1 to 100,
# and write these 100 results of C in memory starting at 0x2020.
```

```
# initialize B
lui $8, 0xFA19
ori $8, $8, 0xE366
# initialize A
addi $9, $0, 1
# initialize value of A at which main function loop should stop
addi $23, $0, 101
# initialize memory address for storing C
ori $10, $0, 0x2020
# loop for hash function data generation, stop branching when $9 == 101
loop:
# A * B
mult $9, $8
# A1 = hi XOR lo
mfhi $11
mflo $12
xor $13, $11, $12
# A1 * B
mult $13, $8
mfhi $11
mflo $12
# A2 = hi XOR lo
xor $13, $11, $12
# A2 * B
mult $13, $8
mfhi $11
mflo $12
# A3 = hi XOR lo
xor $13, $11, $12
# A3 * B
mult $13, $8
```

```

mfhi $11
mflo $12
# A4 = hi XOR lo
xor $13, $11, $12
# A4 * B
mult $13, $8
mfhi $11
mflo $12
# A5 = hi XOR lo
xor $13, $11, $12
# A5[31:16]
srl $14, $13, 16
# A5[15:0]
andi $15, $13, 0x000FFFF
# C = A5[31:16] XOR A5[15:0]
xor $15, $14, $15
# C[15:8]
srl $16, $15, 8
# C[7:0]
andi $17, $15, 0x000000FF
# C = C[15:8] XOR C[7:0]
xor $17, $16, $17
# store C in memory
sw $17, 0($10)
# increment memory address to next word
addi $10, $10, 4
# increment A by 1
addi $9, $9, 1
# loop again with next value of A
bne $9, $23, loop

# among the 100 calculated values,
# search for the largest C and write its address in 0x2000,
# and its value in 0x2004.
# reset address to load from 0x2020 onwards
ori $10, $0, 0x2020
# set target stop address for end of loop_biggest iteration
ori $23, $0, 0x21B0
# initialize first value to be compared
lw $18, 0($10)

# loop for comparisons, stop branching when when stop address is reached
loop_biggest:
# increment memory address to next word
addi $10, $10, 4
lw $19, 0($10)
# set comparison flag if new bigger number is loaded
slt $9, $18, $19
# branch to end of loop if not set
beq $9, $0, not_bigger
# if flag set, record address of new biggest value and move biggest value for next comparison
add $20, $0, $10
add $18, $0, $19
not_bigger:
# loop again with next address
bne $10, $23, loop_biggest

# store address of largest value in 0x2000
ori $10, $0, 0x2000
sw $20, 0($10)
# store largest value in 0x2004
ori $10, $0, 0x2004
sw $18, 0($10)

# check if each C in the array contains "1111" anywhere in the 8 bit binary (Y / N)
# the total number of "Y" should be written into memory location 0x2008.
# reset address read head to 0x2020, target stop address in $23 is already 0x21B0
ori $10, $0, 0x2020
# initialize register containing pattern
addi $21, $0, 31
# reset $9 for use as counter for values that match
add $9, $0, $0

# loop for comparisons, stop branching when when stop address is reached
loop_1111:
# initialize value to be compared
lw $18, 0($10)
# first comparison: check if bits 4:0 of value is 0b11111
andi $22, $18, 31
# skip increment if not equal
beq $22, $21, match_1111
# second comparison: shift value right to compare for bits 5:1
srl $22, $18, 1
andi $22, $18, 31
beq $22, $21, match_1111
# third comparison: shift value right to compare for bits 6:2
srl $22, $18, 2
andi $22, $18, 31
beq $22, $21, match_1111
# fourth comparison: shift value right to compare for bits 7:3
srl $22, $18, 3
andi $22, $18, 31
beq $22, $21, match_1111
# no match, go to end of loop
# (this is supposed to be a j instruction but the 15-instruction limit was reached)
beq $0, $0, endoffloop
# pattern match increment
match_1111:
addi $9, $9, 1
# next loop around
endoffloop:
# increment memory address to next word
addi $10, $10, 4
bne $10, $23, loop_1111

# store match count in 0x2008
ori $10, $0, 0x2008
sw $9, 0($10)

```