NetID: ms591

Mariam Sulakian
Professor Sorin
CS 250: Homework 2

ASSEMBLY PROGRAMMING

Mips Instruction Set

1. 0000 0011 0011 0010 1000 0000 0010 0010 → **sub $s0, $t9, $s2**

Format	Opcode (6)	rs (5)	rt (5)	rd (5)	shamt (5)	func (6)
R-type	0000 00	11 001	1 0010	1000 0	000 00	10 0010
	0	25	18	16	0	34
		$t9	$s2	$s0	0	sub

op: base operation
rs, rt: first and second source registers
rd: destination register
shamt: shift amount, only used in shift instructions
func: used with op to select an arithmetic instruction

2. sw $r7, 12($r4) → **1010 1100 1000 0111 0000 0000 0000 1100**

Format	opcode (6)	rs (5)	rt (5)	imm (16)
I-type	sw	$r4	$r7	12
	1010 11	$a0	$a3	0000 0000 0000 1100
	1010 11	00 100	0 0111	0000 0000 0000 1100

op: base operation
rs: source register (address for loads and stores, or an operand for branch/immediate arith instructs)
rt: source register for branches, but a destination register for the other I-type instructions
imm: immediate

Compiling C Code to MIPS Assembly

3. Using file: Homework 1, prog_part.c

a. How many total instructions (not dot-prefixed directives like .frame) are in each of the two versions?

- Optimization disabled (-O0): 66 instructions
- All performance optimizations enabled (-O3): 33 instructions

b. How many memory accesses (loads and stores) were in each of the two versions?

- Optimization disabled (-O0): 18
- All performance optimizations enabled (-O3): 6

c. Which version of the code uses more registers?

- The all performance optimizations enabled (-O3) version uses more registers.

d. Based on the above, what are some general strategies you suspect the optimizer takes to improve performance?

- To improve performance, the optimizer uses more registers and accesses memory less. With more registers, the function has to put less on the stack. The stack is in physical memory, which is RAM. Compared to accessing registers, accessing memory is much slower. Hence, limiting the number of times a program accesses memory improves performance.

Assembly Language Programming in MIPS *attached on Sakai*

4. fibtimes2.s

5. recurse.s

6. HoopStats.s