Week 5 - MIPS ISA

The Stored Program Concept

- All programs (instructions) are just data represented by combinations of bytes!
- Any block of memory can be code. Consequently, self-modifying code is possible!
- The Program Counter (PC) is a special register (not directly accessible) which holds a pointer to the current instruction.

Instruction Formats

MIPS instructions come in three flavors:

R-Instruction format (register-to-register). *Examples: addu, and, sll, jr*

op code	rs	rt	rd	shamt	funct
6 bits	5 bits	5 bits	5 bits	5 bits	6 bits

See green sheet to see what registers are read from and what is written to

I-Instruction Format (register immediate) Examples: addiu, andi, bne

op code	rs	rt	immediate
6 bits	5 bits	5 bits	16 bits

Note: Immediate is 0 or sign-extended depending on instruction (see green sheet)

J-Instruction Format (jump format) For j and jal

op code	address
6 bits	26 bits

KEY: An instruction is R-Format if the op code is 0. If the opcode is 2 or 3, it is J-format. Otherwise, it is I-format. Different R-format instructions are determined by the "funct".

1. How many instructions are representable with this format?

32 bits gives an upper bound of 2^32 possible instructions. We don't utilize all of the possible permutations though. Counting the possible instructions in each format we get 64 R types (opcode == 0, 6 bits of freedom in func), I - 61 (opcode not 0,2,3), J - (j, jal)

2. What could we do to increase the number of possible instructions?

variable length instructions, larger instruction length, expand opcode at the expense of other fields.

Addressing in MIPS

1. The Program Counter (PC) holds the address of the currently executing instruction
2. **Branch Addressing -** Uses Relative Addressing - beq, bne, bgez, bltz,...
   nextPC = signExtend(immediate<<2) + PC + 4
3. **Memory Addressing -** Uses Base Displacement Addressing - sw, lw, sb, lb,...
   memAddr = signExtend(immediate) + R[rs]
4. **Jump Addressing -** uses Pseudodirect Addressing (Absolute) - j,jal,...
   nextPC = PC[31:28] | zeroExtend(address<<2)
5. **Jump Register Addressing -** Uses Register Addressing - jr
   nextPC = R[rs] (usually, nextPC = $ra)

6. What instruction is 0x00008A02?

0000 0000 0000 0000 1000 1010 0000 0010
000000 00000 00000 10001 01000 000010 - R type
srl $s1, $0, 8

Example MIPS Assembling

Fill in the following table with the correct fields from the MIPS routine below. Then fill in the second table with each instruction translated to the raw 32-bit hex number.

Addr	opcode	rs	rt	rd	shamt	funct
0x00	0	0	0	16	0	0x20
0x04	0	0	0	17	0	0x20
0x08	0x08	0	18			16
0x0c	0x04	16	18			4
0x10	0x23	16	8			0
0x14	0	0	8	17	0	0x20
0x18	0x08	16	16			4
0x1c	0x03					4
0x20	0	0	0	0	0	0x00

Addr	MIPS			Raw Bits (in hex)
0x00		add	$s0, $0, $0	0x00008020
0x04		add	$s1, $0, $0	0x00008820
0x08		addi	$s2, $0, 16	0x20120010
0x0c		beq	$s0, $s2, L2	0x12120004
0x10	L1:	lw	$t0, 0($s0)	0x8e080000
0x14		add	$s1, $0, $t0	0x00088820
0x18		addi	$s0, $s0, 4	0x22100004
0x1c		j L1		0x08000004
0x20	L2:	sll	$0, $0, 0	0x00000000