CS61C Fall 2013 – 3 – The Formation of Integers and MIPS

The Formation of Integers

Idea	Implementation	Pros	Cons
Unsigned	**Start 0 as 0000 0000. Make 1 into 0000 0001. Count upwards.**	**Continuous.**	**No negative numbers.**
Sign/Magnitude	**Sign is first bit (1 = -, 0 = +) Other bits are like unsigned.**	**Has negative numbers.**	**Not continuous.**
One's complement	**If first bit zero, read unsigned. Otherwise, flip all bits and read negative unsigned.**	**Fixed positive slope.**	**2 zeroes. Not continuous.**
Two's complement	**If first bit zero, read unsigned. Otherwise, flip all bits and add 1. Read negative unsigned.**	**Fixed positive slope. 1 Zero.**	**1 more negative number. Not continuous.**

Differences in Representation

Fill in the following table, assume only 5 bit numbers.

Decimal	0	-1	15	-16	MAX	MIN
Unsigned						
Sign / Mag.						
One's Comp.						
Two's Comp.						

Changing Bases

Translate the following:		
To Base 8 10_{10} 77_{10} 64_{7} To Base 16 (Unsigned) 0000 0000 1011 1000 1010 1001 1111 1110	To Base 10 (Binary is in 2's C) 0_{100} 211_{3} $0F_{16}$ $0000\ 0000_{2}$ $0010\ 0000_{2}$ $1000\ 0000_{2}$ $1111\ 1100_{2}$ $1111\ 1111_{2}$	To 8-bit Binary (2's C) 0 15 128 -18 -128 FA_{16} 364_{8} 3213_{4}

Intro to MIPS

The Stored Program Concept

- All programs (instructions) are just data represented by combinations of bytes!
- Any block of memory can be code; self-modifying code possible (it's likely system will protect against this)
- The Program Counter (PC) - special register (not directly accessible), holds a pointer to current instruction.
- For recursion: adjust the stack pointer ($sp) to save return address ($ra) and other registers (ex: $s0)

Instruction	Syntax	Example
Add	add dst, src0, src1	add $s0, $s1, $s2
Add Immediate	addi dst, src0, immediate	addi $s0, $s1, 12
Shift Left Logical	sll dst, src, immediate	sll $t0, $s0, 4
Load Word	lw dst, offset(bAddr)	lw $t0, 4($s0)
Store Word	sw dst, offset(bAddr)	sw $t0, 4($s0)
Branch if not equal	bne src0, src1, brAddr	bne $t0, $t1, notEq
Branch if equal	beq src0, src1, brAddr	beq $t0, $t1, Eq
Jump unconditional	j jumpAddr	j jumpWhenDone
Jump register	jr reg	jr $ra

C To MIPS

C Code	MIPS Code
int a = 5, b = 10; **if (a + a == b) {** ** a = 0;** **} else {** ** b = a - 1;** **}**	

C To MIPS: Recursion

C Code	MIPS Code
int sum(int n) { ** return n ? n + sum(n – 1) : 0;** **}** **// Use recursion in your MIPS!**	

MIPS To C

C Code	MIPS Code
	addi $s0, $0, 0 **addi $s1, $0, 1** **addi $t0, $0, 30** **loop: beq $s0, $t0, done** ** sll $s1, $s1, 1** ** addi $s0, $s0, 1** ** j loop** **done: # done**