CS61C Fall 2012 – 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.**	**Continuous, mostly.**	**2 zeroes.**
Two's complement	**If first bit zero, read unsigned. Otherwise, flip all bits and add 1. Read negative unsigned.**	**Completely continuous. 1 Zero.**	**1 more negative number. Is this really a con / unavoidable?**

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

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.

Instruction Formats

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

MIPS Addressing Modes

- We have several **addressing modes** to access memory (immediate not listed):
  - **Base displacement addressing**: Adds immediate to register value to create memory address (used for lw, lb, sw, sb)
  - **PC-relative addressing**: Uses PC (actually current PC plus four) and adds I-value of instruction (multiplied by 4) to create address (used by I-format branching instructions like beq, bne)
  - **Pseudodirect addressing**: Uses upper four bits of PC and concatenates a 26-bit value from instruction (with implicit 00 lowest bits) to make 32-bit address (used by J-format instructions)
  - **Register Addressing:** Uses the value in a register as memory (jr)

1) Write MIPS code that will do n mod k where n is the first argument and k is the second.

C Code	MIPS Code
int mod(int n, int k) { **}**	

2) Write MIPS code that will do n^k where n is the first argument and k is the second. Do not use MIPS' mult.

C Code	MIPS Code		
int multiply(int n, int k) { **}** **int power(int n, int k) {** **}**			

3) Write MIPS code that will compute the nth Fibonacci number.

C Code	MIPS Code	
int fibonacci(int n) { **}**		