Outline of Lecture

- Simple Binary Addition and Subtraction
- Overflow
- Logical operations

Addition & Subtraction

- Just like in primary school (carry / borrow 1s)

Eamples

0101	0111	1110
+ 0001	- 0110	- 0101
0110	0001	1001

- An alternative for subtraction is to add the negated number

Examples

0111	same as	0111
+ 1010		- 0110
10001		

- Since only 4 bits are available, drop the extra bit on the left most.

- In other words, 0111+1010 gives 0001

Exercise

- Assume 4- bit 2's complement numbers. What are:
  - a) 1001 - 1100 =
  - b) 1100 - 1001 =
  - c) 0010 - 1111 =
  - d) 1100 + 1001 =
  - e) 1101 + 0101 =
- Carry out the followings in 5 - bit 2's complement additions:
  - f) 5 - 10 =
  - g) -4 - 7 =

Overflow

- **The result may be too large for finite computer word. This is called *overflow*.**

  For example, adding two n-bit numbers does not always yield an n-bit number

  0111

  +0001

  1000

  note that overflow term is some what mis-leading, it does not mean a carry overflowed

- **7 + 1 should be 8, but 8 can't be represented using 4 bits.**
- **Notice the sign bits are inconsistent when an overflow occurs, two positive numbers add to give a negative number.**

Overflow

- **The overflow conditions for addition and subtraction are listed below:**

Operation	Operand A	Operand B	Result
A + B	≥ 0	≥ 0	< 0
A + B	< 0	< 0	≥ 0
A - B	≥ 0	< 0	< 0
A - B	< 0	≥ 0	≥ 0

Detecting Overflow

- No overflow when adding a +ve and a -ve number
- No overflow when signs are the same for subtraction
- Overflow occurs when the value affects the sign:
  - overflow when adding two positives yields a negative
  - adding two negatives gives a positive
  - subtract a negative from a positive and get a negative
  - subtract a positive from a negative and get a negative
- Consider the operations A + B, and A - B
  - Can overflow occur if B is 0 ?
  - Can overflow occur if A is 0 ?

Exercise

- In which of the followings would overflow occur?

  Assume 5 - bit 2's complement numbers.

  a) 10011 - 01101

  b) 10010 + 11010

  c) 01001 + 01101

  d) 01100 + 01111

  e) 10000 - 10010

Effects of Overflow

- An exception condition occurs and interrupts normal program execution

  - Control jumps to predefined address for exception

  - Interrupted instruction address is recorded for possible resumption

- Details based on software system / language

  - example : flight control vs. homework assignment

- Don't always want to detect overflow

  - new MIPS instructions treating operands as unsigned numbers: addu, addiu, subu, sltu, sltiu

Logical Operations

- Bitwise AND and bitwise OR

  - perform and /or bit by bit

- 0001 and 1011 = 0001
- 0001 or 1011 = 1011

Examples

- and $t0, $s0, $s1
- or $t0, $s1, $s0
- andi $t0, $s0, 32
- ori $t0, $s2, 15

- Also , XOR, NOR , NOT

More Logical Operations

- It is quite useful in many programs to operate on fields and bits within a word.
- One such operations are called *shifts*. They move all the bits in a word to the left or to the right, filling the emptied bits with 0s.
- MIPS has two shift instructions: *shift left logical* (`sll`) and *shift right logical* (`srl`).

```
sll  $10, $16, 8        # left shift the
  con   tent of $16     by 8 bits and put
  result in $10
```

- if $ 16 contains

  01001010**00000000000110100000000** then

- $10 will contain

  00000000000110100000000000000000

- Note that the 01001010 is lost and the 00000000 is shifted in from nowhere.

More Logical Operations

- **Shift right arthimetic (sra)**

```
sra  $10, $16, 7 # right shift with sign
  extension
```

If $16 contains

0**10010100000000000011010000100 0**

then $10 will contain

00000000**10010100000000000011010**

Note that the 0001000 is lost

If $16 contains

1**10010100000000000011010000100 0**

then $10 will contain

11111111**10010100000000000011010**

It is like dividing the number by 2^7