Arithmetic and Logical Operations

Carryout and Overflow

Example: unsigned arithmetic

Let's say we have an ISA called Tiny ISA where registers are _____

and we have the instruction ADDIU $t1 $t0 1 where:

ADDUI : _____

$t0 : _____ $t1 : _____

Functionally, this instruction means: _____

Say $t0 has the value of _____

What is the result of this instruction?

```
     ___ ___ ___ ___ ___ ___ ___ ___  $t0
 +   ___ ___ ___ ___ ___ ___ ___ ___
_____________________________________
  ___ ___ ___ ___ ___ ___ ___ ___ ___  $t1
```

Carryout

- ➔ When the _____ of an arithmetic computation _____ the _____ we have to store that outcome.

Overflow

- ➔ If the outcome of an arithmetic instruction is _____ due to the _____ of how the value is stored
- ➔ If the addition of two _____ results in a _____
- ➔ If the addition of two _____ results in a _____

Note

➔ Overflow will never occur when adding _____

➔ Carryout and overflow can occur _____

Example: two's complement arithmetic

In the same Tiny ISA, we have an instruction ADDI $t3 $t2 0xFF where $t2 = 0x01
What two's complement number does 0xFF represent? Do we have overflow?

```
     __ __ __ __ __ __ __ __   $t2
 +   __ __ __ __ __ __ __ __
_____________________________

  __ __ __ __ __ __ __ __ __   $t3
```

More examples: 8-bit two's complement arithmetic

Which of these examples have carry out? Which have overflow? Problems were adapted from: http://sandbox.mc.edu/~bennet/cs110/tc/add.html

0xD9 + 0x5C __ __ __ __ __ __ __ __ + __ __ __ __ __ __ __ __ _____ __ __ __ __ __ __ __ __ __ Carry out? Overflow?	0xED + 0xF9 __ __ __ __ __ __ __ __ + __ __ __ __ __ __ __ __ _____ __ __ __ __ __ __ __ __ __ Carry out? Overflow?	0x2C + 0x2D __ __ __ __ __ __ __ __ + __ __ __ __ __ __ __ __ _____ __ __ __ __ __ __ __ __ __ Carry out? Overflow?
0x68 + 0x2D __ __ __ __ __ __ __ __ + __ __ __ __ __ __ __ __ _____ __ __ __ __ __ __ __ __ __ Carry out? Overflow?	0xB5 + 0x3B __ __ __ __ __ __ __ __ + __ __ __ __ __ __ __ __ _____ __ __ __ __ __ __ __ __ __ Carry out? Overflow?	0x99 + 0xBB __ __ __ __ __ __ __ __ + __ __ __ __ __ __ __ __ _____ __ __ __ __ __ __ __ __ __ Carry out? Overflow?

0x0A + 0xFD __ __ __ __ __ __ __ __ + __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ Carry out? Overflow?	0x7F + 0x01 __ __ __ __ __ __ __ __ + __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ Carry out? Overflow?	0xFF + 0x01 __ __ __ __ __ __ __ __ + __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ Carry out? Overflow?

Bitwise Operations

Examples

01001101 AND 01001001 0 1 0 0 1 1 0 1 0 1 0 0 1 0 0 1	01001101 OR 01001001 0 1 0 0 1 1 0 1 0 1 0 0 1 0 0 1
01001101 XOR 01001001 0 1 0 0 1 1 0 1 0 1 0 0 1 0 0 1	NOT 01001101 ~ 0 1 0 0 1 1 0 1
0xAA AND 0xF0 AA: __ __ __ __ __ __ __ __ F0: __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __	0xAA OR 0xF0 AA: __ __ __ __ __ __ __ __ F0: __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __

Reduction Operation

Example: perform the reduction operation on 01001101 and 01001001

AND 01001001 0 • 1 • 0 • 0 • 1 • 0 • 0 • 1 =	XOR 01001101 0 ⊕ 1 ⊕ 0 ⊕ 0 ⊕ 1 ⊕ 1 ⊕ 0 ⊕ 1 =

A ⊕ B ⊕ C

A	B	C	A⊕B	A⊕B⊕C

1 ⊕ 0 ⊕ 1 =

(1 ⊕ 0) ⊕ 1

1 ⊕ (0 ⊕ 1)

Shifts

Logical Shift

Right: replace _____ with _____ Left: replace _____ with _____

Shift 0x5F by 3 bits to the right

___ ___ ___ ___ ___ ___ ___ ___

1 bit ___ ___ ___ ___ ___ ___ ___ ___

2 bits ___ ___ ___ ___ ___ ___ ___ ___

3 bits ___ ___ ___ ___ ___ ___ ___ ___

Shift 0x5F by 3 bits to the left

___ ___ ___ ___ ___ ___ ___ ___

1 bit ___ ___ ___ ___ ___ ___ ___ ___

2 bits ___ ___ ___ ___ ___ ___ ___ ___

3 bits ___ ___ ___ ___ ___ ___ ___ ___

Arithmetic

Right: replace _____ with _____ Left: replace _____ with _____

Shift 0xA7 by 3 bits to the right

___ ___ ___ ___ ___ ___ ___ ___

1 bit ___ ___ ___ ___ ___ ___ ___ ___

2 bits ___ ___ ___ ___ ___ ___ ___ ___

3 bits ___ ___ ___ ___ ___ ___ ___ ___

Shift 0xA7 by 3 bits to the left

___ ___ ___ ___ ___ ___ ___ ___

1 bit ___ ___ ___ ___ ___ ___ ___ ___

2 bits ___ ___ ___ ___ ___ ___ ___ ___

3 bits ___ ___ ___ ___ ___ ___ ___ ___

Rotate

Rotate 0xA6 by 3 bits to the right	Rotate 0xA6 by 3 bits to the left
___ ___ ___ ___ ___ ___ ___ ___	___ ___ ___ ___ ___ ___ ___ ___
1 bit ___ ___ ___ ___ ___ ___ ___ ___	1 bit ___ ___ ___ ___ ___ ___ ___ ___
2 bits ___ ___ ___ ___ ___ ___ ___ ___	2 bits ___ ___ ___ ___ ___ ___ ___ ___
3 bits ___ ___ ___ ___ ___ ___ ___ ___	3 bits ___ ___ ___ ___ ___ ___ ___ ___

Multiplication

Division