Modifications to LC3
Arithmetic & Control
Instructions

Based on slides by Amir Roth
Modified by Diana Palsetia(2009-2010)

Opcode for NOT

Assembly Mnemonic
NOT R2, R6 ; R2 = ~R6

Assembly compatible with LC3 i.e. same mnemonic
Not binary compatible: different encoding
LC3 uses opcode 1001 for NOT

OR

LC3 has no support for OR
- Can synthesize using DeMorgan’s
- Expensive in terms of Registers

Assembly Mnemonic
OR R1, R2, R6 ; R1 = R2 | R6;

SUB, MUL, DIV

CIT 593
**JMP**

Semantics

\[
PC = PC + 1 + \text{SEXT(IMM11)}
\]

(IMM11 = Immediate 11-bit signed value)

Assembly Mnemonic

JMP <label>

Different from LC3

- LC3 JMP is in a register (can completely specify an address)
- More like BRnzp except the PCOffset it 11 bits compared to 9-bits

**CMPI**

Semantics

\[
\text{val} = \text{regs}[SR1] - \text{SEXT(IMM7)}
\]

sets NZP condition codes

Assembly mnemonic

CMI R1, #1

Why not just use ADD(Immediate)?

- CMPI doesn’t have an output to register

**Multication Example Revisited**

for(C = 0; B > 0; B--) { C += A;}

SUB R3, R3, R3 ; C = 0

LOOP

; test
CMPI R2, #0
BRz DONE

if(B < 0) {
  B = -B;
  A = -A;
}

ADD R3, R3, R1 ; body
ADD R2, R2, #-1 ; re-init
JMP LOOP ; go back to LOOP

DONE

**More Compare Operations**

CMP and CMPI perform signed comparison

i.e. xFFFF < x0000

CMPU and CMPIU perform unsigned comparison

i.e. xFFFF > x0000

Useful for comparing addresses