
CPSC / ECE 3710, Aug 31

Microprocessor Design Lab
Ganesh Gopalakrishnan

MCR16-ts

- LUI -- Load Upper Immediate -- load a byte into the upper byte of a word
 - ORI -- OR a register with an immediate data item
 - LOAD -- Load from a memory address into a register
 - ADD -- Add the contents of two registers, storing the result into a third
 - STOR -- Store a register into memory
-

MCR16-ts Program

- `lui 0xBF r1 ; r1 = BF00 now`
 - `ori 0xFF r1 ; OR FF to r1; r1 = BFFF now`
 - `load r1 r1 ; r1 <- contents(BFFF)`
 - `lui 0xC0 r2 ; r2 = C000 now`
 - `load r2 r2 ; r2 <- contents(C000)`
 - `add r1 r2 ; r2 <- contents(r1) + contents(r2)`
 - `lui 0xC0 r3 ; r3 = C000 now`
 - `ori 0x01 r3 ; r3 = C001 now`
 - `stor r2 r3 ; Location C001 <- contents(r2)`
-

Machine Instruction Format

- OPPOSITE Rdest ImmHi/OpExt ImmLo/Rsrc
 - 15-12 11-8 7-4 3-0
-

MCR16-ts Program: LUI

- LUI Imm, Rdst
 - Binary (Machine Code): 1111 Rdst ImmHi ImmLo
 - BEHAVIOR:
 - $\text{let } va = \text{bv_lshift}(\text{zex}(\text{Imm}, 16), 8)$
 - $\text{RF}' = \text{RF}[\text{Rdst}] := va$
 - $\text{PC}' = \text{PC} + \text{zex}(1, 16)$
-

MCR16-ts Program: ORI

- ORI Imm, Rdest
 - Machine Code: 0010 Rdst ImmHi ImmLo
 - let vd = RF[Rdst]
 - and vs = zex(Imm, 16)
 - and va = bv_or(vd,vs)
 - RF' = RF[Rdst] := va
 - PC' = PC + zex(1, 16)
-

MCR16-ts Program: LOAD

- LOAD Rdest, Raddr
 - 0100 Rdest 0000 Raddr
 - let va = MEM [RF[Raddr]]
 - RF' = RF[Rdest] := va
 - PC' = PC + zex(1, 16)
-

MCR16-ts Program: ADD

- ADD Rsrc, Rdest
 - 0000 Rdst 0101 Rsrc

 - let vd = RF[Rdst]
 - let vs = RF[Rsrc]
 - let va = vd + vs
 - let carry = cb(vd+vs)
 - let ovflo = if (msb(vd) XOR msb(vs)) then 0 -- recall that this is how
 - else (msb(vs) XOR msb(va)) -- overflow is computed

 - then

 - RF' = RF[Rdst] := va,
 - F' = ovflo,
 - C' = carry,
 - PC' = PC + zex(1, 16)

MCR16-ts Program: STOR

- STOR Rsrc, Raddr
 - 0100 Rsrc 0100 Raddr
 - let vd = RF[Raddr]
 - and vs = RF[Rsrc]
 - $MEM' = MEM [vd] := vs$
 - $PC' = PC + zex(1, 16)$
-

MCR16-bl instructions (some of them)

- ADDI Imm, Rdst
 - 0101 Rdst Immhi Immlo

 - let $vd = RF[Rdst]$
 - and $vs = \text{sex}(Imm, 16)$
 - and $va = vd + vs$
 - and $\text{carry} = \text{cb}(vd+vs)$
 - and $\text{ovflo} = \text{if } (\text{msb}(vd) \text{ XOR } \text{msb}(vs)) \text{ then } 0$
□ $\text{else } (\text{msb}(vs) \text{ XOR } \text{msb}(va))$

 - $RF' = RF[Rdst] := va,$
 - $F' = \text{ovflo},$
 - $C' = \text{carry},$
 - $PC' = PC + \text{zex}(1, 16)$

MCR16-bl : SUB

- SUB Rsrc, Rdst

- 0000 Rdst 1001 Rsrc

- let $vd = RF[Rdst]$

- and $vs = RF[Rsrc]$

- and $va = vd - vs$

- and $carry = cb(vd-vs)$

- and $ovflo = \text{if } (msb(vd) \text{ XNOR } msb(vs)) \text{ then } 0$

- $\text{else } (msb(vs) \text{ XNOR } msb(va))$

- $RF' = RF[Rdst] := va,$

- $F' = ovflo,$

- $C' = carry,$

- $PC' = PC + zex(1, 16)$

MCR16-bl : SUB (alternative defn)

- SUB Rsrc, Rdst
 - 0000 Rdst 1001 Rsrc

 - let $vd = RF[Rdst]$
 - and $vs = twosc(RF[Rsrc])$
 - and $va = vd + vs$
 - and $carry = cb(vd+vs)$
 - and $ovflo = \text{if } (msb(vd) \text{ XOR } msb(vs)) \text{ then } 0$
 - $\text{else } (msb(vs) \text{ XOR } msb(va))$

 - $RF' = RF[Rdst] := va,$
 - $F' = ovflo,$
 - $C' = carry,$
 - $PC' = PC + zex(1, 16)$

MCR16-bl : SUBI

- SUBI Imm, Rdst
 - 1001 Rdst Immhi Immlo
 - let $vd = \text{RF}[\text{Rdst}]$
 - and $vs = \text{sex}(\text{Imm}, 16)$
 - and $va = vd - vs$
 - and $\text{carry} = \text{cb}(vd - vs)$
 - and $\text{ovflo} = \text{if } (\text{msb}(vd) \text{ XNOR } \text{msb}(vs)) \text{ then } 0$
 - else $(\text{msb}(vs) \text{ XNOR } \text{msb}(va))$
 - $\text{RF}' = \text{RF}[\text{Rdst}] := va,$
 - $\text{F}' = \text{ovflo},$
 - $\text{C}' = \text{carry},$
 - $\text{PC}' = \text{PC} + \text{zex}(1, 16)$

MCR16-bl : CMP

- CMP Rsrc, Rdst
 - 0000 Rdst 1011 Rsrc
 - let $vd = RF[Rdst]$
 - and $vs = RF[Rsrc]$
 - and $va = vd - vs$
 - and $carry = cb(vd-vs)$
 - and $neg = carry \text{ XOR } msb(vd) \text{ XOR } msb(vs)$
 - and $zero = va = zex(0, 16)$
 - $L' = carry$
 - $N' = neg$
 - $Z' = zero$
 - $PC' = PC + zex(1, 16)$

How will you program this algorithm

- Given three numbers, find which is the max and output that info (details on class webpage)

