

MIPS instructions

Lecture B09



Lecture notes section B09

2002-01-18

CSE1303 Part B lecture notes

1

Last time

- MIPS R2000 architecture
 - memory organization
 - registers
- Running programs
 - fetch-execute cycle
 - SPIM simulator

2002-01-18

CSE1303 Part B lecture notes

2

In this lecture

- MIPS R2000 instructions
 - instruction set
 - format
 - encoding
 - pseudoinstructions
- Sign
 - signed vs unsigned instructions
 - sign/zero extending

2002-01-18

CSE1303 Part B lecture notes

3

MIPS instruction set

- MIPS programs made up of instructions
- Instruction set is all possible instructions that can be performed by the MIPS processor
 - all programs are made up of instructions selected from the instruction set

2002-01-18

CSE1303 Part B lecture notes

4

MIPS instruction set

- Arithmetic instructions
 - add, sub, mul, div, rem, neg
- Logic instructions
 - and, or, xor, nor, not
 - sll, srl, sra
- Data movement instructions
 - li, move

2002-01-18

CSE1303 Part B lecture notes

5

MIPS instruction set

- Load/store instructions
 - lb, lh, lw, la, sb, sh, sw
- Control transfer instructions
 - j, jr, jal, jalr
 - beq, bne, blt, b1e, bgt, bge
- Miscellaneous instructions
 - syscall
- Other instructions
 - not a complete set, but all you'll need for CSE1303

2002-01-18

CSE1303 Part B lecture notes

6

Arithmetic instructions

- MIPS instructions exist for
 - addition (+)
 - add \$t0, \$t1, \$t2 # \$t0 = \$t1 + \$t2
 - subtraction (-)
 - sub \$t0, \$t1, \$t2 # \$t0 = \$t1 - \$t2
 - multiplication (*)
 - mul \$t0, \$t1, \$t2 # \$t0 = \$t1 * \$t2
 - division (/)
 - div \$t0, \$t1, \$t2 # \$t0 = \$t1 / \$t2
 - remainder (%)
 - rem \$t0, \$t1, \$t2 # \$t0 = \$t1 % \$t2
 - negate (unary -)
 - neg \$t0, \$t1 # \$t0 = -\$t1

2002-01-18

CSE1303 Part B lecture notes

7

Bitwise logic instructions

- MIPS instructions exist for
 - bitwise AND (&)
 - and \$t0, \$t1, \$t2 # \$t0 = \$t1 & \$t2
 - bitwise OR (|)
 - or \$t0, \$t1, \$t2 # \$t0 = \$t1 | \$t2
 - bitwise exclusive OR (XOR) (^)
 - xor \$t0, \$t1, \$t2 # \$t0 = \$t1 ^ \$t2
 - bitwise not-OR (NOR)
 - nor \$t0, \$t1, \$t2 # \$t0 = ~(\$t1 | \$t2)
 - bitwise NOT (unary ~)
 - not \$t0, \$t1 # \$t0 = ~\$t1

2002-01-18

CSE1303 Part B lecture notes

8

Shift instructions

- MIPS instructions exist for
 - shift left (logical) (<<)
 - fill with zero bits
 - sll \$t0, \$t1, 5 # \$t0 = \$t1 << 5
 - shift right (logical) (>>)
 - fill with zero bits
 - srl \$t0, \$t1, 5 # \$t0 = \$t1 >> 5
 - shift right (arithmetic) (>>)
 - fill with copies of MSB
 - sra \$t0, \$t1, 5 # \$t0 = \$t1 >> 5

2002-01-18

CSE1303 Part B lecture notes

9

Data movement instructions

- MIPS instructions exist for
 - move (copy) value between registers
 - move \$t0, \$t1 # \$t0 = \$t1
 - load (copy) immediate (constant) value into register
 - li \$t0, 5 # \$t0 = 5

2002-01-18

CSE1303 Part B lecture notes

10

Load instructions

- MIPS instructions exist for
 - load (read) byte from memory to GPR
 - lb \$t0, var # \$t0 = (signed char) var
 - load (read) halfword (16 bits) from memory to GPR
 - lh \$t0, var # \$t0 = (short) var
 - load (read) word from memory to GPR
 - lw \$t0, var # \$t0 = (long) var
 - "load" (compute) memory address, without dereferencing
 - la \$t0, var # \$t0 = &var

2002-01-18

CSE1303 Part B lecture notes

11

Store instructions

- MIPS instructions exist for
 - store (write) byte from GPR to memory
 - sb \$t0, var # var = (char) \$t0
 - store (write) halfword from GPR to memory
 - sh \$t0, var # var = (short) \$t0
 - store (write) word from GPR to memory
 - sw \$t0, var # var = (long) \$t0

2002-01-18

CSE1303 Part B lecture notes

12

Jump instructions

- MIPS instructions exist for
 - jump (go) to label
 - `j foo` # go to foo
 - jump to label and link (remember origin)
 - `jal foo` # \$ra = here; go to foo
 - jump to address contained in register
 - `jr $t0` # go to address at (\$t0)
 - jump (and link) to address contained in register
 - `jalr $t0` # \$ra = here; go to (\$t0)

2002-01-18 CSE1303 Part B lecture notes 13

Branch instructions

- MIPS instructions exist for
 - branch (go) to label if one value is [?] another
 - equal to
 - `beq $t1, $t2, foo` # if \$t1 == \$t2 goto foo
 - not equal to
 - `bne $t1, $t2, foo` # if \$t1 != \$t2 goto foo
 - less than
 - `blt $t1, $t2, foo` # if \$t1 < \$t2 goto foo
 - less than or equal to
 - `ble $t1, $t2, foo` # if \$t1 <= \$t2 goto foo
 - greater than
 - `bgt $t1, $t2, foo` # if \$t1 > \$t2 goto foo
 - greater than or equal to
 - `bge $t1, $t2, foo` # if \$t1 >= \$t2 goto foo

2002-01-18 CSE1303 Part B lecture notes 14

Miscellaneous instructions

- MIPS instructions exist for
 - operating system service call (for I/O)
 - `syscall`

2002-01-18 CSE1303 Part B lecture notes 15

MIPS instruction format

- All MIPS instructions occupy 32 bits (4 bytes)
- Instruction contains
 - opcode (operation code)
 - specifies type of instruction
 - operands
 - values or location to perform operation on
 - registers
 - immediate (constant) numbers
 - labels (addresses of other lines of program)

2002-01-18 CSE1303 Part B lecture notes 16

MIPS instruction format

- Three general assembler formats

`sub $t0, $t1, $t2` R (for "register") format instruction: three registers
subtract the contents of register \$t2 from the contents of register \$t1; put the result in register \$t0

`addi $v0, $a0, 42` I (for "immediate") format instruction: two registers and one immediate operand
add the immediate number 42 with the contents of register \$a2; put the result in register \$v0

`j foo` J (for "jump") format instruction: one label (address) operand
 jump (go) to the line with the label foo and continue running from there

2002-01-18 CSE1303 Part B lecture notes 17

MIPS instruction format

- Instruction's components encoded in binary
 - opcode and operands
 - must fit in 32 bits

opcode determines how remaining bits are to be interpreted as operands

001000 00100 00100 1111111111111111

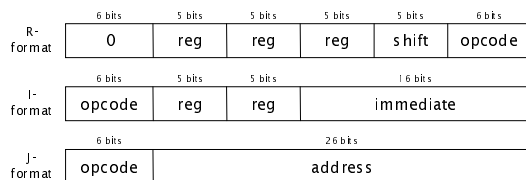
source register destination register immediate value

opcode (6 bits): 001000, (8₁₀) means "add immediate"

2002-01-18 CSE1303 Part B lecture notes 18

MIPS instruction format

- Three general encoding formats
 - depend on instruction's operand types
 - register, immediate and/or address



2002-01-18

CSE1303 Part B lecture notes

19

Immediate instructions

- Binary (two-operand) operations can apply to two registers
 - use R-format instruction format
 - add \$t0, \$t1, \$t2 # \$t0 = \$t1 + \$t2
- Binary operations can apply to one register and one immediate (constant) value
 - use I-format instruction format
 - add \$t0, \$t1, 123 # \$t0 = \$t1 + 123
- Immediate value is always second value (not counting destination register)
 - add \$t0, 123, \$t1 # not valid

2002-01-18

CSE1303 Part B lecture notes

20

Immediate instructions

- Instructions with immediate forms
 - addi, andi, ori, xori
 - also (sort of) lb, lh, lw, sb, sh, sw
 - other immediate instructions are synthesized by assembler (pseudoinstructions)
- Can write instruction as immediate form explicitly
 - addi \$t0, \$t1, 123 # Real instruction
- Or (easier) can allow assembler to figure out instruction form
 - add \$t0, \$t1, 123 # Turned into addi

2002-01-18

CSE1303 Part B lecture notes

21

Pseudoinstructions

- To help programmer, assembler provides some instructions which are not actually part of machine
 - e.g., move, li, la
- These instructions are expressed in terms of other, real, instructions
 - move \$t0, \$t1 ⇔ or \$t0, \$t1, \$0
 - li \$t0, 5 ⇔ ori \$t0, \$0, 5
 - la \$t0, -8(\$fp) ⇔ addiu \$t0, \$fp, -8

2002-01-18

CSE1303 Part B lecture notes

22

Pseudoinstructions

- Instructions are sometimes assembled to others, depending on operands
 - add \$t0, \$t1, 24 ⇔ addi \$t0, \$t1, 24
 - sub \$t0, \$t1, 24 ⇔ addi \$t0, \$t1, -24
- Some pseudoinstructions are assembled to sequences of instructions
 - blt \$t1, \$t2, foo ⇔
 - slt \$at, \$t1, \$t2 # set if less than
 - bne \$at, \$zero, foo
- Handled transparently by assembler
 - programmer can usually ignore details

2002-01-18

CSE1303 Part B lecture notes

23

Extension

- addi is I-format instruction
 - 16 bits available for immediate value
 - registers are 32 bits wide
- Can't add 16-bit value to 32-bit value
 - words must be same width
 - must convert 16-bit immediate value to equivalent 32-bit one
 - converted by extending 16-bit value
- Extension required any time value is moved from a narrower word into a wider one

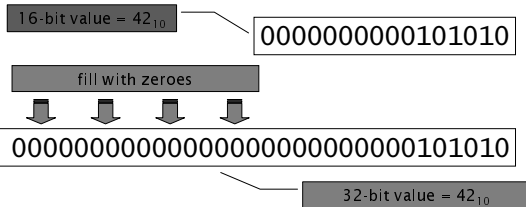
2002-01-18

CSE1303 Part B lecture notes

24

Zero extension

- Unsigned values can be extended by adding zeroes in front



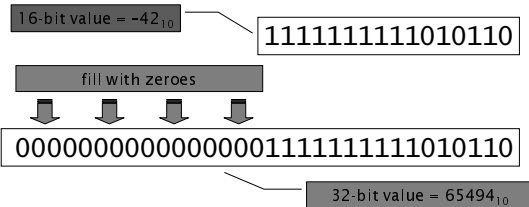
2002-01-18

CSEI 303 Part B lecture notes

25

Sign extension

- Zero extension does not work for negative signed values



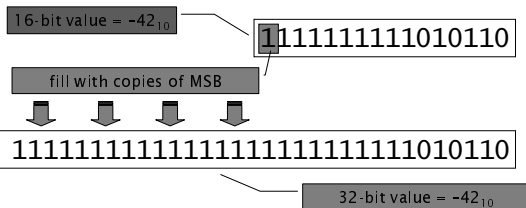
2002-01-18

CSEI 303 Part B lecture notes

26

Sign extension

- For signed numbers, sign extend by duplicating MSB (sign bit)



2002-01-18

CSEI 303 Part B lecture notes

27

Extension

- Sign or zero extension performed on immediate field of I-format instructions
 - `addi`, `lbu`, `lh`, `lw`, `sb`, `sh`, `sw` perform sign extension
 - `andi`, `ori`, `xori` perform zero extension

2002-01-18

CSEI 303 Part B lecture notes

28

Unsigned instructions

- Many instructions have unsigned versions
 - suited for calculations with unsigned values
 - MIPS treats registers as containing unsigned values
 - for some operations, unsigned version ignores overflow
 - for some operations, unsigned version zero-extends instead of sign-extends
 - most unsigned instructions formed by appending `u` to opcode

2002-01-18

CSEI 303 Part B lecture notes

29

Unsigned instructions

- Instructions with unsigned forms
 - `mulu`, `divu`, `remu`, `bltu`, `bltu`, `bgtu`, `bgeu`
 - treat operands as unsigned
 - `addu`, `addiu`, `subu`, `negu`
 - ignore overflow
 - `lbu`, `lhu`
 - zero-extend loaded value (`lbu` and `lhu` sign-extend)
 - `srl`
 - shift right, fill with zeroes (more appropriate for unsigned values than `sra`)

2002-01-18

CSEI 303 Part B lecture notes

30

A MIPS program

```
# Program to convert inches to millimetres.
# Integer approximation, multiply by 254/10.

# Data used by the program
.data
inch: .word 42      # Allocate word, store value 42
mm:   .space 4     # Allocate word, value unimportant

# Program starts from label main.
.text
main: lw $t0, inch  # get inches into $t0
      mulu $t1, $t0, 254 # multiply...
      divu $t2, $t1, 10 # ...and divide
      sw $t2, mm      # store result in mm

      li $v0, 10     # system call 10...
      syscall        # ...exits simulation
```

2002-01-18

CSE1303 Part B lecture notes

31

Covered in this lecture

- MIPS R2000 instructions
 - instruction set
 - format
 - encoding
 - pseudoinstructions
- Sign
 - signed vs unsigned instructions
 - sign/zero extending

2002-01-18

CSE1303 Part B lecture notes

32

Going further

- Complete instruction set
 - SPIM manual
- Floating-point instructions
 - SPIM manual
- 64-bit instructions
 - for MIPS R4000 processor

2002-01-18

CSE1303 Part B lecture notes

33

Next time

- Programming in MIPS
- Comparing C with MIPS



Reading:
Lecture notes section B1.0

2002-01-18

CSE1303 Part B lecture notes

34

Copyright

Copyright © 2001 Deborah Pickett.
No part of this presentation may be
duplicated without permission from
the author.

2002-01-18

CSE1303 Part B lecture notes

35