中山大學
SUN YAT-SEN UNIVERSITY

国家超级计算广州中心
NATIONAL SUPERCOMPUTER CENTER IN GUANGZHOU

Compiler Design
编 译 器 构 造 实 验

Lab 12: Project 4

张献伟

xianweiz.github.io

DCS292, 5/27/2021

MIPS: Overview

- MIPS (Microprocessor without Interlocked Pipelined Stages)[无内部互锁流水级的微处理器]
  - 一套精简指令集(RISC)架构，首次发布于1985
  - 2021年宣布停止开发，转换到RISC-V

- Load/Store architecture
  - Only load and store instructions can access memory
  - all other instructions access only registers
    - E.g., all arithmetic and logical operations involve only registers (or constants that are stored as part of the instructions)

MIPS: Registers

- The MIPS R2000 CPU has 32 registers
  - 31 of these are general-purpose registers that can be used in any of the instructions
  - The last one, denoted register zero, is defined to contain the number zero at all times
  - Even though any of the registers can theoretically be used for any purpose, there are guidelines that specify how each of the registers should be used

Symbolic Name	Number	Usage
zero	0	Constant 0.
at	1	Reserved for the assembler.
v0 - v1	2 - 3	Result Registers.
a0 - a3	4 - 7	Argument Registers $1 \cdots 4$.
t0 - t9	8 - 15, 24 - 25	Temporary Registers $0 \cdots 9$.
s0 - s7	16 - 23	Saved Registers $0 \cdots 7$.
k0 - k1	26 - 27	Kernel Registers $0 \cdots 1$.
gp	28	Global Data Pointer.
sp	29	Stack Pointer.
fp	30	Frame Pointer.
ra	31	Return Address.

MIPS: Instruction Set

- Arithmetic
  - E.g., add dst, src1, src2 // dst = src1 + src2
- Comparison
  - E.g., sge des, src1, src2 // des ← 1 if src1 ≥ src2, 0 ow
- Branch and jump
  - E.g., bge src1, src2, lab // branch to lab if src1 ≥ src2
  - E.g., j, lab // jump to lab
- Load, store, and data movement
  - E.g., lw des, addr // load the word at addr into des
  - E.g., move des, src1 // copy the contents of src1 to des

SPIM: A MIPS32 Simulator

- Reads/executes assembly source programs
  - Does not execute binaries
  - Unlike a real MIPS assembler, SPIM has no support for string literals in the text segment, so it is necessary to use a .data
- Usages
  - *load "src0.s"*: to load a program into the simulator
    - Note that program name must be enclosed in quotes
  - *run*: to run the program
    - By default, execution begins with the statement labeled 'main'
  - *quit*: to terminate and return to shell

```
$ ./spim.linux
SPIM Version 6.0 of July 21, 1997
Copyright 1990-1997 by James R. Larus (larus@cs.wisc.edu).
All Rights Reserved.
See the file README for a full copyright notice.
Loaded: ./trap.handler
(spim) load "src1.s"
(spim) run
x>=0(spim) quit
```

Example

```
/* Ex1: Assignment statement */
program ex1;
class c1
{
   declarations
      int x=-1;
   enddeclarations
   method void main()
   declarations
      int x=4;
   enddeclarations
   {
     if (x>=0)
     {
       system.println('x>=0');
     };
   }
}
```

```
.data
Enter:   .asciiz "
"
base:
.text
.data
V0:
        .word   -1
.text
main:
        la      $t1     V0
        sw      $t1     0($sp)
        addi    $sp     $sp     -4
        sw      $fp     0($sp)
        addi    $sp     $sp     -4
        move    $fp     $sp
        sw      $ra     0($sp)
        addi    $sp     $sp     -4
        li      $t1     4
        sw      $t1     0($sp)
        addi    $sp     $sp     -4
        add     $t3     $fp     -4
        sw      $t3     0($sp)
        addi    $sp     $sp     -4
        li      $t2     0
        lw      $t3     4($sp)
        addi    $sp     $sp     4
        add     $t1     $t3     $t2
        lw      $t1     0($t1)
        sw      $t1     0($sp)
        addi    $sp     $sp     -4
        li      $t1     0
        sw      $t1     0($sp)
        addi    $sp     $sp     -4
        lw      $t1     4($sp)
        addi    $sp     $sp     4
        lw      $t2     4($sp)
        addi    $sp     $sp     4
        sge     $t1     $t2     $t1
        sw      $t1     0($sp)
        addi    $sp     $sp     -4
        lw      $t1     4($sp)
        addi    $sp     $sp     4
        beqz    $t1     L1
.data
V1:     .asciiz "x>=0"
.text
        li      $v0     4
        la      $a0     V1
        syscall
        b       L0
L1:
L0:
        lw      $ra     0($fp)
        move    $sp     $fp
        lw      $fp     4($sp)
        addi    $sp     $sp     4
        addi    $sp     $sp     4
        jr      $ra
```