Tutorial 7

EduMIPS64

Course project

- Project website
- Worth 30% of your grade
- Milestones
  - Project proposal (5%): due Oct. 31
  - Progress report (10%): due Nov. 16
  - Final presentation (25%): due Dec. 5/6
  - Final report (60%): due Dec. 6
- Programming involved
- Could take a while! Start **early**

EduMIPS64

- Free (open-source) MIPS64 simulator
  - Only the integer pipeline is implemented
- Written in Java
- Compiled using Apache ANT
- Runs assembly programs
  - Subset of MIPS64 assembly (and no FP)
- Basic I/O facilities
  - Read/write to/from the console
  - Read/write to/from files

Obtaining the software

- Download the latest source release
  - Sourceforge link
  - edumips64-0.5.3.tar.bz2
- Extract the sources
  - tar xvjf edumips64-0.5.3.tar.bz2
- Build the sources
  - ant
- Run the simulator
  - java -jar edumips64-svn.jar

Tested on
tr5130gu-1.ece.mcgill.ca

Using EduMIPS64

- Read the manual!
- Runs programs written in MIPS64 assembly
- Will not run assembly generated by a (cross-)compiler (e.g.: gcc)
- Shows
  - Timing diagrams, pipeline state
  - Register file / memory contents
  - RAW hazards (can enable forwarding)
  - CC / instructions processed / CPI
- Allows you to step through the code and see the resulting changes on the processor state

MIPS64 assembly

```
; This is a comment
.data
label: .word 15     ;This is an inline comment

.code               ;or .text
daddi   r1, r0, 1
syscall 0           ; TRAP 0; HALT
```

MIPS64 assembly

- Two sections
  - Data: where you store the data on which you compute (corresponds to memory)
    - Can store bytes (1B), half-words (2B), words (4B) or double-words (8B)
  - Code: there your actual program instructions live
    - 32 integer registers (R0-R31)
    - Operands can be
      - `Registers [ R1 ]`
      - `Immediate values [ 10 ]`
      - `Addresses [ 0(R1) ]`

MIPS64 assembly

- Instructions can be
  - ALU instructions
    - AND, DADD, DADDU, DDIV, ...
  - Load/store instructions
    - LD, SD, ...
  - Flow control instructions
    - J, JR, B, BEQ, ...
  - System calls
    - exit, open, close, read, printf, ...
  - Other
    - **BREAK**, NOP, **HALT**

Dinero Frontend

- Behavioral cache simulator
    - Allows you to test different cache strategies
    - Reads trace files, which list memory addresses (I/R/W)

```
...
i 000000000000001C 4
i 0000000000000020 4
r 00000000000000D8 8
w 0000000000000170 8
...
```

Code example

When writing assembly,
start with C!

- **Have a look at those samples**

  http://www.edumips.org/attachment/wiki/Upload/Samples-pack-0.1.tar.bz2

Code structure

- Directories
  - **core/**: main code of the simulator
  - data/: some documentation
  - docs/: main documentation
  - libs/: external libraries used by the simulator
  - ui/: user interface
  - utils/: exception classes, translations, user settings
  - test/: validation test programs

Code structure

- core/: main code of the simulator
  - Parser.java: parse the assembly source file
  - BitSet{32,64}.java: represent a 32/64-bit quantity
  - CPU.java: top-level entity representing the processor
    - SymbolTable.java: map labels to data/instructions
    - Register.java: single GPR element (R0 special case)
    - Memory.java: data and instruction memories
      - MemoryElement.java: one data memory element
      - is/Instruction.java: one instruction
  - IOManager.java: process open/read/... system calls

Code structure

- core/is/: instructions definitions
  - Instruction.java: generic definition of an instruction
    - ALUInstructions.java: ALU
      - ALU_{I,R}Type.java
        - ADDU.java, ...
    - FlowControlInstructions.java: Flow control
      - FlowControl_{I,J,R}Type.java
        - BNE.java, ...
    - LDSTInstructions.java: Load/Store
      - Loading.java, Storing.java
        - LD.java, ...
  - InstructionUtils.java: define binary ALU operations

Testing your changes

- The modifications you make to the simulator must <u>not</u> impact the outcome of any program
- You are expected to test your modifications thoroughly
  - Determine what might break
  - Design test programs to validate those cases
- You may also validate your changes against the unmodified version of the simulator
- See test/ for sample test programs

Source version control

- It is recommended that you use a version control system for your coding work
- **Git** and **mercurial** are widely used nowadays
  - Pro Git - e-Book
  - Hg Init: a Mercurial tutorial
- You can host your repositories online for easy collaboration. e.g.:
  - Bitbucket (free private repo., supports git/mercurial)
  - Github (free public repo., supports git)