Introduction to Computer Architecture
Project 1

MIPS Binary Code Read

Hyungmin Cho
Department of Software
Sungkyunkwan University

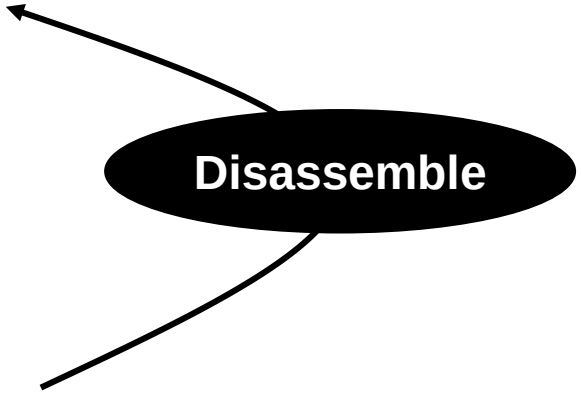
Project Requirement

- Your program reads a binary file filled with MIPS machine code, and print the assembly representation of the code

- Not a full simulator yet..

Assembly language program (MIPS)

```
swap:
      muli $2, $5,4
      add  $2, $4,$2
      lw   $15, 0($2)
      lw   $16, 4($2)
      sw   $16, 0($2)
      sw   $15, 4($2)
      jr   $31
```



Machine (object, binary) code (MIPS)

```
00000000101000010000000000011000
00000000000011000000110000100001
10001100011000100000000000000000
10001100111100100000000000000100
10101100111100100000000000000000
10101100011000100000000000000100
00000011111000000000000000001000
```

Test Sample

- You can obtain test input files from the following location
  - `~swe3005/2020s/proj1/test1.bin`
  - `~swe3005/2020s/proj1/test2.bin`

```
00000000: 0022 0020 8d42 0020 2230 0008 1440 0004
00000010: 0000 0000 03e0 0008 0000 0000 a7c4 0008
00000020: 0013 5940 0000 000d
```

Test Result

- The expected results files are in the following location
  - ~swe3005/2020s/proj1/test1.txt
  - ~swe3005/2020s/proj1/test2.txt

```
inst 0: 00220020 add $0, $1, $2
inst 1: 8d420020 lw $2, 32($10)
inst 2: 22300008 addi $16, $17, 8
inst 3: 14400004 bne $2, $0, 4
inst 4: 00000000 sll $0, $0, 0
inst 5: 03e00008 jr $31
inst 6: 00000000 sll $0, $0, 0
inst 7: a7c40008 sh $4, 8($30)
inst 8: 00135940 sll $11, $19, 5
inst 9: 0000000d unknown instruction
```

Program Interface

- Your program should provide a simple shell
  - Print a prompt on each line “`mips-sim>` ”
  - Accept a line of user command
  - Loop until the program exits

- Need to support the following two commands
  - read <filename>
    - Read the binary file named <filename> and prints the disassembled instruction
    - Each line prints in the following format

```
inst <instruction number>: <32-bit binary code in hex format> <disassembled instruction>
```

  - exit
    - Finish the shell loop and exit

Execution Results

```
$ ./mips-sim
mips-sim> read test1.bin
inst 0: 00220020 add $0, $1, $2
inst 1: 8d420020 lw $2, 32($10)
inst 2: 22300008 addi $16, $17, 8
inst 3: 14400004 bne $2, $0, 4
inst 4: 00000000 sll $0, $0, 0
inst 5: 03e00008 jr $31
inst 6: 00000000 sll $0, $0, 0
inst 7: a7c40008 sh $4, 8($30)
inst 8: 00135940 sll $11, $19, 5
inst 9: 000000d unknown instruction
mips-sim> exit
$
```

Disassemble Format

- Instruction name in lowercase
  - `add, sub, sw, jal,` etc…
- Registers are all represented in numbers
  - `$0, $1, $20`, …
  - Do not to use their name (`$s0, $t2`, etc…)
- Immediate and address values are represented in signed decimal
  - `sw $16, 20($29)`
  - `addi $29, $29, -16`

Instructions to support

- `add, addu, and, div, divu, jalr, jr, mfhi, mflo, mthi, mtlo, mult, multu, nor, or, sll, sllv, slt, sltu, sra, srav, srl, srlv, sub, subu, syscall, xor, addi, addiu, andi, beq, bne, lb, lbu, lh, lhu, lui, lw, ori, sb, slti, sltiu, sh, sw, xori, j, jal`

- If there is an instruction that can't be interpreted, print "unknown `instruction`" as the disassembled format

Things to Consider

- Endianness!
  - Input file (e.g., **test.bin**) uses the big endian format
  - Your computer uses the little endian format

- Shift instructions

Project Rule

- You can use any language you'd like to use, but it must be compliable and executable on the department server

- You need to provide a `Makefile` to compile/execute your code
  - `make`
    - Compile your program
    - Do nothing if you're using a script language
  - `make run`
    - Execute your program
  - `make clean`
    - Erase all generated files (executable, object file, etc…)

Makefile Example

- C

```
CC=gcc
CCFLAGS=

#add C source files here
SRCS=main.c

TARGET=mips_sim

OBJS := $(patsubst %.c,%.o,$(SRCS))

all: $(TARGET)

%.o:%.c
	$(CC) $(CCFLAGS) $< -c -o $@

$(TARGET): $(OBJS)
	$(CC) $(CCFLAGS) $^ -o $@

.PHONY=clean
.PHONY=run

run: $(TARGET)
	./$(TARGET)

clean:
	rm -f $(OBJS) $(TARGET)
```

- Python

```
all:

.PHONY=clean
.PHONY=run

run:
	python mips_sim.py

clean:
```

Project Environment

- We will use the department's In-Ui-Ye-Ji cluster
  - `swin.skku.edu`
  - `swui.skku.edu`
  - `swye.skku.edu`
  - `swji.skku.edu`
  - ssh port: 1398

- First time users (CS & SW department)
  - ID: your student ID (e.g., 2019123456)
  - Use the default password
    - CS & SW departments: your last name in uppercase: e.g., HONG)
    - Other departments: same as your student ID
  - MUST change your password after the first login (Use **`yppasswd`** command)

- http://cs.skku.ac.kr/news/notice/view/2587

Submission

- Clear the build directory
  - Do not leave any executable or object file in the submission

- Use `submit` program
  - If you want to submit "`src`" directory…
    - `~swe3005/bin/submit proj1 src`

```
Submitted Files for proj1:
File Name                                 File Size       Time
--------------------------------------------------------------------------
proj1-2019123456-Sep.05.17.22.388048074        268490                  Thu Sep  5 17:22:49 2019
```

- Verify the submission
  - `~swe3005/bin/check_submission proj1`

Project 1 Due Date

- 2020 Apr 15th, 23:59:59

- Late penalty (max 3 days): 20% per day

Addendum

- Input rule
  - Maximum number of characters per line (including command and newline) is 100 bytes
    - You can make your program to take more than 100 characters, but we won't test for it.
  - There is only one space character between the command and filename, and there is no space before the command
  - The filename does not contain any space