

CSI 402 – Spring 2011
Programming Assignment IV

Administrative Information

- **Deadline:** 11 PM, Wednesday, Apr. 13, 2011.
Cutoff: 11 PM, Friday, Apr. 15, 2011.
- The program must have two or more C source files.
- All the files (C source files, header files and the `makefile`) must be submitted together using the `turnin-csi402` command.
- README file
 `~csi402/public/prog4/prog4.README`
will be available by 10 PM on Tuesday, Mar. 29, 2011.
- The README file will contain information regarding `turnin-csi402` and additional specifications for the `makefile`.

Project Description

- **Goal:** To write an assembler for a hypothetical computer called TMIPS (“Tiny MIPS”).
- May use a one-pass or a two-pass design.
- **Important Requirement:** Symbol table *must* be implemented as a hash table of size 13 using the hash function shown in Example 1 of Handout 6.1.

Weightage: 10%

Total Points: 100 (Correctness: 85, Str. & doc: 15).

Caution

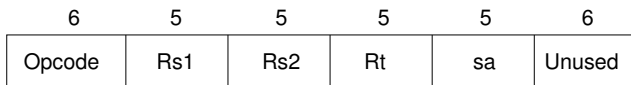
This is a challenging project. You are advised to start working on the project right away.

Brief Information about TMIPS

- Memory size = $2^{16} = 65,536$ words.
- Word size = 32 bits; uses **little endian** numbering of bits within a word.
- Instruction length = 32 bits.
- 32 registers each of size 32 bits. (Registers are denoted by \$0, ..., \$31.)
- Supports only the `int` data type; 32-bit integers (2's complement form for negative integers).
- **Addressing modes:** Direct, Base + Displacement and Immediate operand.

TMIPS Instruction Formats

(a) R-Format:

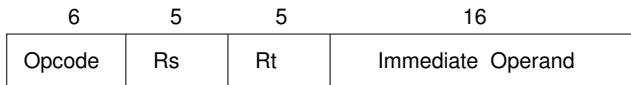


Examples:

```
add    $3,$2,$1
move   $5,$3
prrr   $21
```

TMIPS Instruction Formats (continued)

(b) I-Format:



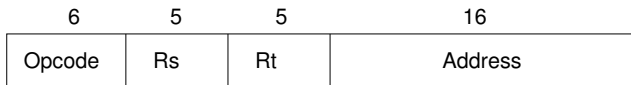
Note: Immediate operand is in 2's complement form with bit 15 as sign bit.

Examples:

```
addi    $3,$2,-7
swb     $5,-2($4)
```

TMIPS Instruction Formats (continued)

(c) J-Format:



Note: Address treated as a 16-bit unsigned integer.

Examples:

```
j          next
jgt       $7,$9,loop
lwa       $5,val
```

TMIPS Assembly Language

- Line has at most 80 characters (including the newline character).
- Blank lines allowed.
- Comments start with '#'.
 - Fields of an instruction:
 - Label (optional); if present, terminated by ':'.
(Conditions for valid labels given in the handout.)
 - Opcode.
 - Operands (optional).
 - Inline comment (optional – also begins with '#').
- List of machine opcodes: See handout.
- List of pseudo-opcodes (directives): `.text`, `.data`, `.resw` and `.word`.
- If both data and text segments are present, data segment comes after the text segment.

Example: A TMIPS Assembly Language Program

```
        .text

begin:  lwa    $5,arr    # $5 has -7.
        swa    $5,x1    # Store -7 at x1.

#Some more instructions.

        move   $7,$0
        add    $6,$7,$5
        prr    $6
        hlt

#Data segment begins here.

        .data

x1:     .resw  10
arr:    .word  -7:15
```

Note: See handout for additional details.

- Starting address of the program is zero.
- Locations created using `.resw` must be initialized to zero.
- Unused bits in an instruction must be set to zero.

Errors to be Detected:

- Errors in the assembly language program: Undefined label(s), multiply defined label(s) and illegal opcodes. (These are reported in the error file as discussed later.)
- Usual Unix command line errors. (These error messages must be written to `stderr` and the program must stop right away.)

```
% p4 infile
```

- p4: Executable version of your program.
- *infile*: Name of input file containing a TMIPS assembly language program.
- Program must produce an object file when there are no syntax errors in the source program; otherwise, an error file must be produced.
- Naming convention for output files: To be discussed in class. (Details also appear in the handout.)

Object File Format:

- Object file is a text file.
- Each line has address and contents (both hexadecimal values) separated by a spaces or tabs.

Error File Format

- Error file is also a text file.
- **First part:** A source listing consisting of each line of source file (including comments and blank lines) preceded by a line number (starting with 1).
- **Second part:** For each line containing an error, a line number followed by a brief error message.
- Error messages must have line numbers in **increasing** order.
- Lists of multiply defined symbols and undefined symbols (if any) must be given after all the error messages.
- **Assembler cannot stop at the first error.** It must find errors in all the lines of the source file.
- If a line has two or more errors, report any one of the errors.

Additional Remarks and Suggestions

Additional Remarks:

- Your program must produce either the object file (when there are no errors in the source file) or the error file.
- There is no concept of a byte in TMIPS. Successive words have addresses 0, 1, 2, . . . , 65535 (decimal).
- In any assembly language instruction specifying registers, the first specified register is the target register, referred to as Rt in the handout. (This is important to correctly assemble the instructions.)

Suggestions:

- Do the assembly in memory, that is, use an `int` array of size 65536 (decimal) to store assembled instructions.
- When assembly is complete, write the contents of the array appropriately to the object file.

Suggestions (continued)

- For each of the three instruction formats, have a separate C source file that contains all the functions needed to assemble statements for that format.
- Review hash tables.
- Review the use of `strtok` from `string.h`.
- You will need to use a number of bitwise operations in generating the object file.

Examples of Assembly: To be presented in class.