

CSI 333 – Fall 2011
Programming Assignment IV

- **Team Project.**
- **Deadline:** 11 PM, Friday, Nov. 11, 2011.
Cutoff: 11 PM, Sunday, Nov. 13, 2011.
- The C source file for the project must be named `p4.c`.
- The source file must be submitted using the `turnin-csi333` command.
- **Each team must make only one submission.**
- README file (on `itsunix.albany.edu`) by 10 PM on Tuesday, November 2, 2011.
`~csi333/public/prog4/prog4.README`

Important Remarks:

- Programs that don't compile or don't generate the executable won't receive any credit.
- Your program must compile and work correctly on `itsunix.albany.edu`.

Lateness Policy:

- No penalty if the program is submitted **by 11 PM on Friday, Nov. 11, 2011**.
- Lateness penalty: 10 points per day.
- Program **won't** be accepted **after 11 PM on Sunday, Nov. 13, 2011**.
- If you submit both a regular version and a late version, only the late version will be graded.

Project Description

Goal: Producing a listing and/or a cross reference table for a program in MIPS Assembly Language (MAL).

Weightage: 10%

Total Points: 100

For students working individually:

Correctness: 85 points

Str. & doc.: 15 points

For students working in a team:

Correctness: 65 points

Str. & doc.: 15 points

Team work: 20 points

Team Projects: Additional Information

- Each team member **must** participate in developing, documenting and testing the program.
- Each team should include additional documentation at the beginning of the source file indicating how the work for the project was divided between the two team members. (Indicate clearly who developed each function and how the testing work was divided between the team members.)
- After the submission deadline, each team must meet with their TA. During the meeting, the TA will ask questions about the team's program and determine the points for team work. (The two team members may receive different scores for team work.)

Program Description

Note: Assume that the executable version of your program is in the file named `prog4.out`.

Command Line Details:

```
unix2> prog4.out  flag  infile  outfile
```

Explanation:

- The arguments *infile* and *outfile* specify the names of the input and output files respectively.
- The input file contains a program written in MAL.
- The output file must contain a listing of the MAL program with line numbers and/or a cross reference table for the MAL program.
- The *flag* argument dictates what the output file must contain.

Valid *flag* Arguments:

- -l : Indicates that the output file must contain *only* the program listing.
- -c : Indicates that the output file must contain *only* the cross reference table (CRT).
- -b : Indicates that the output file must contain *both* the program listing and the CRT.

Details Regarding MAL Programs

- Each line of a MAL program has at most 80 characters (including the newline character).
- Blank lines are 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 '#').
- Maximum number of labels = 100

Details Regarding MAL Programs (continued)

Example:

```
loop:   sw    $15,avg    #Store reg. 15
```

- In the above instruction, the label is `loop`, the opcode is `sw` and the operands are `$15` and `avg`.
- The instruction **defines** the identifier `loop` and **uses** the identifier `avg`.

Program Listing: Contains each line of the input program with a line number.

Cross Reference Table (CRT): For each identifier in the MAL program, the CRT specifies the line number where the identifier is defined and the lines where it is used.

A Sample Input File

Name of input file: mal_prog.txt

```
.data
x1:    .word    80
x2:    .word    0
      .text
begin: lw      $15,x1
      add     $15,$15,x2
      sw      $15,x1
end:   done
```

- Suppose we execute the following Unix command:

```
unix2> prog4.out -b mal_prog.txt output.txt
```

- The contents of the file `output.txt` are shown on the next slide.

Contents of the Output File (output.txt)

```
1      .data
2  x1:  .word   80
3  x2:  .word   0
4      .text
5  begin: lw      $15,x1
6         add     $15,$15,x2
7         sw      $15,x1
8  end:  done
```

Cross Reference Table

Identifier	Definition	Use
x1	2	5 7
x2	3	6
begin	5	
end	8	

Important Notes

- Study the additional details regarding MAL programs given on page 2 of the handout.
- Study the longer example on pages 3 and 4 of the handout carefully to understand all the requirements.

Errors to be Detected:

- Command line errors (wrong number of command line arguments, invalid flag or one of the specified files can't be opened).
- If any of the above errors occur, print a suitable message to `stderr` and exit.

Programming Suggestions

- **Data structure to be used:** Array (of size 100) of struct, where each struct has the following data members:
 - A char array of size 11 (to store an identifier),
 - an integer (to store the line number where the identifier is defined) and
 - a pointer to a linked list; each node of the list stores an integer for the line number where the identifier is used.
- Don't assume any limit on the number of lines in the input file.
- Read the input line by line (using `fgets`).
- For each line of the input file which is not a comment line or a blank line, use `strtok` to parse the line (i.e., extract the various fields of the instruction).

- Program must use command line arguments.
- Your program must contain several functions in addition to `main`.
- **Note:** In a MAL program, an identifier may be used before it is defined.

Program Grading and Other Notes

- Programs will be graded using a script written by the TAs.
- The script will compile your source program, generate the executable version and run the executable on new test data.
- The TAs will grade the version that you submit; once the submission is closed, you won't be allowed to make any changes to your program.
- You must follow the programming and documentation guidelines indicated in "Course Policies".