

COMP 3050-202 Computer Architecture

Homework #6 Spring, 2018

- This assignment is due no later than midnight (11:59:59 PM) **April 19/20 (M/T)**.
- **YOU WILL FIND A WORKING MICROCODE FILE WITH SUPPORT FOR THE MULT, DIV AND RSHIFT INSTRUCTIONS FROM ASSIGNMENT 4, AS WELL AS A WORKING MASM WITH THE SAME UPDATED SUPPORT AT:**
 - `~bill/cs305/prom_mrd.dat`
 - `~bill/cs305/masm_mrd`
- **All** of your submissions must include a minimum of **four** separate files:
 - **File 1:** A short **write-up** that **first** specifies what you think your **degree of success** with a project is (**from 0% to 100%**), followed by a brief discussion of your approach to the project along with a **detailed description** of any problems that you were **not** able to resolve for this project. **Failure to specifically provide this information will result in a 0 grade** on your assignment. If you do **not disclose** problems in your write-up and problems are detected when your program is tested, you will receive a grade of 0. **Make sure that you include your email address in your write-up so that the corrector can email you your grade.**
 - **File(s) 2(a, b, c, ...):** Your **complete source code** For this assignment you should have a single linker.c source file, the set of masm (.asm) files specified in this handout, and the set of translated object files produced by the masd_mrd -o program, for each .asm file.
 - **File 3:** We don't need a makefile here, since you only have to build the linker from a single .c file, but you should briefly describe the steps needed to build and run this problem in a simple text file, or include a makefile.
 - **File 4:** A file that includes your **resulting output** run(s) from your project. This is a simple text file that shows your output, but make sure that you **annotate it** so that it is self descriptive and that all detailed output is well identified.
- The files described above should be the only files placed in one of your subdirectories, and this subdirectory should be the target of your submit command, this time specifying "hw6". See the on-line file **Assignment_Submit_Details.pdf** for specific directions.

- The problem you must solve has been described in class and is formalized as follows:
- You will provide a new program which will operate as a **linker of Mic1 macroassembly programs**, using the object formats that are output by the current macroassembler (**masm_mrd**) when this tool is used with the **-o** command line option. Your linker program must:
 - Accept an arbitrary number of object files, **passed as command line arguments**, where each object file was built using **the -o option** by the current **masm_mrd** assembler
 - Create a fully resolved, **single executable image** which can be used directly with the Mic1 execution emulator (you will need to use the promfile that includes assignment 3 support)
 - Create an **unresolved but consolidated object file** (exactly like **masm_mrd -o** would) when using this linker with the **-o** option on the command line with multiple previously assembled object files. This consolidated object file could be used later by the linker **to further link it with other object files into an executable**.
 - The output for both situations (**executable or consolidated object file**) should be written to **stdout** by the linker, so that it can be shown directly on the control terminal or redirected to a file for later use
- You must also **re-write the code** you used for assignment #5 into a collection of functions, **each in its own separate .asm file**, with the following signatures:
 - **void main()** the main routine for the program
 - **int readint()** this routine must read a positive-only integer from the keyboard, and return it
 - **void writeint(int binary_int)** this routine must take a single, positive-only binary integer as an argument, and write the integer to the output display, including **nl** and **cr** characters at the end
 - **void writestr(void *strptr)** this routine must take a single argument in the form of the **address of a memory location**, and print the entire string starting at that location to the output display, and also print a **nl** and **cr** character out at the end of the string
 - **int addints(int arg1, int arg2)** this function must take two arguments, each a positive only integer, and compute the sum of the two integers and **return the sum or return a 2s complement value of -1** if the addition would overflow
 - **void xbsywt()** when this function returns, it is safe to store an ascii character into the transmitter buffer (location 4094)
 - **void rbsywt()** when this function returns, it is safe to load an ascii character from the receiver buffer (location 4092)
- Each of the assembly files required above must now be assembled separately, using the **masm_mrd -o** form of the assembler to produce an unresolved output object file
- You must now use your **linker** program to link each of the previously created, unresolved output object files into a single executable, and **you must run** this executable for **each pair of numbers** used in assignment #5

- You must also show that your **linker will work correctly with a `-o` flag** by creating a consolidated object file from the same set of assembled files used to create the executable above
- As outlined above, your submission must include:
 - **File 1:** A short **write-up** that describes **your success level** with this assignment, and **clearly states** what parts of this assignment you were **not** able to do, if any
 - **Files 2:** • The source listings of **each individual assembly file** (you should put all the listings in a single text file)
 - The unresolved object file listings of each individual assembly file after each has been assembled by **`masm_mrd -o`** (you should put all the listings in a single text file)
 - The **C code** for your linker
 - **File 3:** A **makefile** or simple text file to build your linker executable
 - **Files 4:**
 - The **output screen session** generated when using the linked executable with the required data sets (a text file).
 - The **consolidated object file** listing generated by the linker when linking all the previously generated object files using the **`-o`** switch to the linker
- The linker is not required to deal with **duplicate labels**, so make sure that each of your separate assembly files uses unique labels (e.g. if one file uses the label `c1:` to label a location with a value of 1 in it, and you need a value of 1 in some other file, use a different label such as `c1a`:
- For further assistance, see Prof. Moloney's help files:
http://www.cs.uml.edu/~bill/cs305/Assignment_6_help_dir/

A SIMPLE MAIN PROGRAM

```
bash-2.05$ cat main1.asm
```

```
main: lodd arg1:
      push
      lodd arg2:
      push
      call myadd:
      stod rslt:
      halt
      .LOC 10
arg1: 25
arg2: 75
rslt: 0
```

A SIMPLE EXTERNAL FUNCTION

```
bash-2.05$ cat myadd.asm
```

```
myadd: lodl 1
       addl 2
       addd bias:
       retn
bias: 100
```

ASSEMBLE WITH -o OPTION

```
bash-2.05$ ./masm_mrd -o < main1.asm > main1.obj
```

```
bash-2.05$ cat main1.obj
```

```
0  U00000000000000000000 arg1:
1  11110100000000000000
2  U00000000000000000000 arg2:
3  11110100000000000000
4  U11100000000000000000 myadd:
5  U00010000000000000000 rslt:
6  1111111111000000
10 0000000000011001
11 0000000001001011
12 0000000000000000
4096 x
    rslt:      12
    arg2:      11
    arg1:      10
    main:      0
```

ALSO ASSEMBLED WITH -o OPTION

```
bash-2.05$ ./masm_mrd -o < myadd.asm > myadd.obj
```

```
bash-2.05$ cat myadd.obj
```

```
0  10000000000000001
1  10100000000000010
2  U001000000000000000 bias:
3  111110000000000000
4  0000000001100100
4096 x
    bias:      4
    myadd:     0
```

LINK WITH -o TO CONSOLIDATED OBJECT

```
bash-2.05$ ./linker -o main1.obj myadd.obj > cons_link.obj
```

```
bash-2.05$ cat cons_link.obj
```

```
0  U000000000000000000 arg1:
1  111101000000000000
2  U000000000000000000 arg2:
3  111101000000000000
4  U111000000000000000 myadd:
5  U000100000000000000 rslt:
6  1111111111000000
10 0000000000011001
11 0000000001001011
12 0000000000000000
13 1000000000000001
14 1010000000000010
15 U0010000000000000 bias:
16 1111100000000000
17 0000000001100100
```

```
4096 x
```

```
myadd:    13
bias:     17
main:      0
arg1:     10
arg2:     11
rslt:     12
```

LINK TO A FULL EXECUTABLE

```
bash-2.05$ ./linker main1.obj myadd.obj > cons_link.exe
bash-2.05$ cat cons_link.exe
```

Executable Content

```
00000000000001010
11110100000000000
00000000000001011
11110100000000000
11100000000001101
00010000000001100
11111111110000000
11111111111111111
11111111111111111
11111111111111111
00000000000011001
00000000001001011
00000000000000000
10000000000000001
10100000000000010
00100000000010001
11111000000000000
00000000001100100
```

Corresponding code

```
main: lodd arg1:
      push
      lodd arg2:
      push
      call myadd:
      stod rslt:
      halt
      .LOC 10

arg1: 25
arg2: 75
rslt: 0
myadd: lodl 1
      addl 2
      addd bias:
      retn
bias: 100
```