CS 241 – Week 5 Tutorial

MERL

Spring 2015

Summary

- MERL to ASM translation
- Unlinking MERL files

Problems

1. .merl files have three parts:
   - Header
     - In the hexdump above the first line is the cookie: 0x10000002
     - The next line is the length: 0x60
     - The last header line is the code length: 0x20
   - Code: From the code length in the header, we know the next 5 lines are the code body
   - Footer - which contains zero or more REL, ESD, and ESR table entries

     REL Contains two words: the format code (0x01) and the location where the relocatable value is located

     ESD Contains: the format code (0x05), the (relocatable) value of the symbol is to be encoded, the number of characters in the symbol name, and each character encoded in ASCII

     ESR Contains: the format code (0x11), the (relocatable) value of the defined symbol, the number of characters in the symbol name, and each character encoded in ASCII.

   Let's annotate the hexdump with the various parts of the MERL file. Note that `xxd` translates to ASCII for us.

```
0000000: 1000 0002  .... ; cookie
0000004: 0000 0060  ...` ; length
0000008: 0000 0020  ...  ; code length
000000c: 0000 001c  .... ; code
0000010: 0000 0014  ....
0000014: 0000 0014  ....
0000018: 0000 0020  ...
000001c: 0000 0000  ....
0000020: 0000 0001  .... ; REL format code
```

```
0000024: 0000 0014  .... ; location
0000028: 0000 0001  .... ; REL format code
000002c: 0000 000c  .... ; location
0000030: 0000 0011  .... ; ESR format code
0000034: 0000 001c  .... ; location
0000038: 0000 0003  .... ; length of symbol name
000003c: 0000 0064  ...d
0000040: 0000 0065  ...e
0000044: 0000 0066  ...f
0000048: 0000 0005  .... ; ESD format code
000004c: 0000 0014  .... ; location
0000050: 0000 0003  .... ; length of symbol name
0000054: 0000 0061  ...a
0000058: 0000 0062  ...b
000005c: 0000 0063  ...c
```

We can now clean the output up and put it into a format that `printmerl` would produce.

```
cookie   10000002
length         60
clen           20
0000000c       1c
00000010       14
00000014       14
00000018       20
0000001c        0
REL            14
REL             c
ESR            1c def
ESD            14 abc
```

In the original asm code a ESR corresponds to a `.word label` at the location in question and a `.import label` somewhere in the code.

A ESD corresponds to a label definition at the location in question and a `.export label` somewhere in the code.

A REL corresponds to a `.word label` at the location in question. This `.word label` will have been translated into a value which corresponds to the location where this label is defined.

For the ESRs and ESDs we know the label symbol name, for the RELs we don't know what the original symbol name was. For the RELs we will simply use a made up, not already used, symbol name.

Every other line is either a `.word value` or some other MIPS instruction.

We can use these rules to translate the `printmerl` output to something similar to the original asm code.

```
.export abc
.import def
.word label1
.word 0x14
abc: label2: .word label2
.word 0x20
label1: .word def
```

2. Let's begin annotating the given `printmerl` output with the lines of asm code that we know (the lines from main.merl).

   For the lines of asm code in lib.merl that we don't know we can annotate them using the same rules that we used for the previous question.

```
; header
cookie   10000002
length         90
clen           30
; lib's code
0000000c       24     ; .word label4
00000010       18     ; def: .word label3
00000014       14     ; .word 0x14
00000018       24     ; .word label5
; main's code
0000001c       2c     ; .word label1
00000020       14     ; .word 0x14
00000024       24     ; abc: label2: .word label2
00000028       20     ; .word 0x20
0000002c       10     ; label1: .word def
; footer
REL            10
REL            1c
REL            24
REL            2c
REL             c
REL            18
ESD            24 abc
ESD            10 def
```

   There are no ESRs in combined.merl, this is because they have all been resolved. When a ESR is resolved the correct value is placed at the appropriate location and the ESR entry in the table is replaced with a REL. How do we know if any of RELs in combined.merl were the result of a ESR in lib.merl being resolved? This occurs when the value at the addresses of any of the RELs point to a location in main's code. Let's note that these addresses have known labels:

```
; header
cookie   10000002
length         90
clen           30
; lib's code
0000000c       24     ; .word abc
00000010       18     ; def: .word label3
00000014       14     ; .word 0x14
00000018       24     ; .word abc
; main's code
0000001c       2c     ; .word label1
00000020       14     ; .word 0x14
00000024       24     ; abc: label2: .word label2
00000028       20     ; .word 0x20
0000002c       10     ; label1: .word def
; footer
```

```
REL             10
REL             1c
REL             24
REL             2c
REL              c      ; resulted from ESR resolution of abc
REL             18      ; resulted from ESR resolution of abc
ESD             24 abc
ESD             10 def
```

Now we remove main's code from the MERL file and also remove any ESDs or RELs that point to main's code.

```
cookie   10000002
length          xx
clen            xx
0000000c         0      ; .word abc
00000010        18      ; def: .word label
00000014        14      ; .word 0x14
00000018         0      ; .word abc
REL             10
REL              c      ; resulted from ESR resolution of abc
REL             18      ; resulted from ESR resolution of abc
ESD             10 def
```

We can now replaced the RELs that resulted from ESRs with the original ESRs. The values of these locations should also be reset to 0 as we don't know where they point. We can also calculate the new lengths.

```
cookie   10000002
length          6c
clen            1c
0000000c         0      ; .word abc
00000010        18      ; def: .word label
00000014        14      ; .word 0x14
00000018         0      ; .word abc
REL             10
ESR              c abc
ESR             18 abc
ESD             10 def
```

Finally we can now produce the something similar to the original asm file using the same technique used in the previous question.

```
.import abc
.export def
.word abc
def: .word label3
.word 0x14
label3: .word abc
```