Linker Handout

This handout is intended to accompany material covered during lectures and is not considered a replacement for lectures. Dissemination of this handout (including posting on a website) is explicitly prohibited unless permission is obtained. Please report any errors to nanaeem@uwaterloo.ca.

Example

The following table shows two MERL files to be linked together. The MERL on the left has a ESR for the import whereas the MERL on the right has an ESD for the export.

```
RELOCATION/EXTERNAL SYMBOL TABLE CODES:
0x01 = relocate, 0x05 = external definition, 0x11 = external reference

LINKABLE ASSEMBLY LANGUAGE:
.import proc                        | .export proc
lis $1                              | proc:
.word proc                          | jr $31
jalr $1                             |

MERL:
; assembly lang.      ; machine l. | ; assembly lang.      ; machine l.
beq $0, $0, 2         ; 0x10000002 | beq $0, $0, 2         ; 0x10000002
.word endmodule       ; 0x00000034 | .word endmodule       ; 0x0000002c
.word endcode         ; 0x00000018 | .word endcode         ; 0x00000010
                                   |
      lis $1          ; 0x00000814 |       proc:           ;
use1: .word 0 ;for now; 0x00000000 |       jr $31          ; 0x03e00008
      jalr $1         ; 0x00200009 |                       ;
endcode:              ;            | endcode:              ;
                                   |
.word 0x11 ; ext. ref ; 0x00000011 | .word 0x05 ; ext. def ; 0x00000005
.word use1 ; location ; 0x00000010 | .word proc ; location ; 0x0000000c
.word 4    ; length   ; 0x00000004 | .word 4    ; length   ; 0x00000004
.word 112  ; 'p'      ; 0x00000070 | .word 112  ; 'p'      ; 0x00000070
.word 114  ; 'r'      ; 0x00000072 | .word 114  ; 'r'      ; 0x00000072
.word 111  ; 'o'      ; 0x0000006f | .word 111  ; 'o'      ; 0x0000006f
.word 99   ; 'c'      ; 0x00000063 | .word 99   ; 'c'      ; 0x00000063
endmodule:            ;            | endmodule:            ;
```

After linking, the resulting MERL file is:

Assembly	Address	Machine language	Explanation
beq $0, $0, 2	0x00000000	0x10000002	
.word endmodule	0x00000004	0x00000040	
.word endcode	0x00000008	0x0000001c	
lis $1	0x0000000c	0x00000814	
use1: .word 0x18	0x00000010	0x00000018	current value of proc
jalr $1	0x00000014	0x00200009	
proc:			
jr $31	0x00000018	0x03e00008	
endCode:			
.word 0x05	0x0000001c	0x00000005	
.word proc	0x00000020	0x00000018	current value of proc
.word 4	0x00000024	0x00000004	
.word 112	0x00000028	0x00000070	
.word 114	0x0000002c	0x00000072	
.word 111	0x00000030	0x0000006f	
.word 99	0x00000034	0x00000063	
,word 0x01	0x00000038	0x00000001	reloc entry for use1
.word use1	0x0000003c	0x00000010	address of use1
endmodule:			
	0x00000040		

Linker Algorithm

Input: 2 MERL files (m1 and m2)
Output: Single MERL file with m2 linked after m1

```
a <- m1.codeLen - 12
relocate m2.code by a
add a to every address in m2.symbolTable
if m1.exports.label ∩ m2.exports.label ≠ ∅ then ERROR
for each <addr1, label> in m1.imports
    if ∃<addr2, label> in m2.exports
        m1.code[addr1] <- addr2
        remove <addr1, label> from m1.imports
        add addr1 to m1.relocates
for each <addr2, label> in m2.imports // addr2 is already incremented by a
    if ∃<addr1, label> in m1.exports
        m2.code[addr2] <- addr1
        remove <addr2, label> from m2.imports
        add addr2 to m2.relocates
imports = m1.imports ∪ m2.imports
exports = m1.exports ∪ m2.exports
relocates = m1.relocates ∪ m2.relocates

output MERL cookie
output m1.codeLen + m2.codelen + total(import, export, relocates) + 12
output m1.codeLen + m2.codelen + 12
output m1.code
output m2.code
output imports, exports, relocates
```