CS 241 - Lecture 7

Graham Cooper

May 27th, 2015

Creating MERL Files

Relocation tool: CS241.merl

- input: merl file and relocation address
- output: non-relocatable mips file with merl header and footer removed, ready to load at the given address, not 0

mips.twoints, mips.array - optional second argument which is the load address

EG: myobj.merl to be loaded at 0x1000

```
java cs241.merl 0x1000 < myobj.merl > myobj.mips
java mips.twoints myobj.mips 0x1000
```

Loader relocation algorithm

```
read() //skip cookie
endMod <- read() // end of Merl file
codeLen <- read() - 12 // subtract header length = length of code
a <- findFreeRAM(codeLen + Stack)
for(i = 0; i < codeLen; i+= 4)
Mem[a+i] <- read()
i <- codeLen + 12
while(i < endMod)
format <- read()
if(format == 1)
rel <- read() // address to be relocated
Mem[a + rel - 12] += ()a - 12) (forward by alpha length
 - backwards by the header length)
else ERROR
i += 8
```

Linkers

Convenient to split large MIPS programs into smaller ones
- reusable libraries
- team development

Issue - how can the assembler resolve a reference to a label if the label is in a different file?

Solution 1

Concatonate (cat) the asm files together, assemble the result

```
cat a.asm b.asm c.asm | java cs241.binasm > abc.mips
```

This does work, But... Why should we reassemble everything if only one file is updated?

Can we assemble first and then cat?

Issues:
- binaries need to be relocatable
- at most one of hte pieces can be at 0x00
- so we'd be creating MERL $\implies$ this does not produce MERL

Solution 2

We need a tool that understands MERL files and puts them together intelligently - a linker

But still: What should the assembler do with reference to labels that aren't there?

Change the assembler:
When the assembler encounters .word id (Where label id: is not found) it outputs 0x00000000, and indicates that the value of id is needed before the

program can run.

a.asm:

```
lis $3
.word x
```

b.asm:

```
x: ...
...
...
```

a.asm cannot be exectuted until the value of x is known.
How does the assembler notify us?
- Make an entry into the MERL file (more later)

But - we have lost an error check because of the example below, what if we meant abc when we wrote abd?
Eg:

```
lis $3
.word abd
...
...
...
abc: ...
...
...
```

Assembler will not flag an error, it will just ask for abd to be linked in. How can the assembler know what is an error and what is intentional?

New assembler directive:
use the .import id
- Tells the assembler to ask for id to be linked in.
- Does not assemble to a word of mips

So when the assembler sees .word abc:
If abc not present and no import then we error.

MERL Entry:
Format code 0x11 means External Symbol Reference (ESR)

What information should be recorded?
- Name of the symbol
- Where was the symbol used (ie. address of the blank word 0x00000000 to be filled in)

0.0.1 ESR Entry:

```
Word 1 - 0x11\\
word 2 - location where the symbol is used\\
word 3 - Length of the name (by characters, n)\\
word 4 - \\
word 5 - The rest of these are the ASCII chracters in the symbol's name\\
...\\
word 3 + n\\
```

The other side:
a.asm

```
.import abc
lis $3
.word abc
```

external reference to abc

b.asm

```
abc:
sw $4, -4($30)
...
jr $31
```

abc is the beginning of a procedure

c.asm

```
...
abc:
add $1, $1, $2
...
beq $2, $0, abc
```

abc is just an internal label (loop)

How does the linker know which abc to link to?
Can't assume labels won't be duplicated.

How can we make abc in b.asm accessible, and abc in c.asm inaccessible?

Solution: Another assembler directive and MERL Entry

Directive:
.export abc - makes abc available for linking
- does not assemble to a word of MIPS
- tells the assembler to make an entry in the MERL file

External Symbol Definition (ESD):

```
word 1 - 0x05 - formatcode
word 2 - address the symbol represents
word 3 - length of the name (n)
word 4 -
word 5 - name in ASCII (each word is a seperate char)
...
...
word (3 + n) -
```