CS 241 — Lecture 7

Bartosz Antczak *Instructor: Kevin Lanctot* *January 24, 2017*

7.1 Loaders

We know how to convert an assembly language program to machine language (via an assembler), but what happens when we *execute* machine code? When we execute (or run) a program, we store it into primary storage (RAM). We do this because we can read from RAM much faster than we can read from the hard drive (which is where the program is stored).
The **loader** is the program responsible for loading other programs into primary storage and preparing them for execution.

7.1.1 Breaking down the Process of Loaders

Loaders execute a program in using the following series of steps:

1. Determine length of program
2. Allocate RAM starting at an arbitrary address α for the code and a stack (and possibly a heap)
3. Copy the program from secondary storage (e.g., hard drive) into primary storage (RAM) starting at α
4. Load the address α into some register, say \$3
5. Do any work at the end that is required by the program (e.g., `twoints` will print out all the register values)

Machine code is just a sequence of bits, so how do we know which words are addresses that must be adjusted? The answer is that we don't know. We must augment the machine code with information about which words need adjusting if the code is relocated.

7.1.2 MERL

MERL (**M**IPS **E**xecutable **R**elocation **L**inkable file) is an executable binary file that also contains a table containing relocation and linking information. MERL has three parts

- A header
- The MIPS machine code
- The relocation information

The three parts are outlined in detail:

The MERL Header

The header consists of three words (for a total of 12 bytes)

1. **Cookie:** identifies the type of file (its value is 0x1000 0002, which is equivalent to `beq $0, $0, 2`)
2. **File Length:** the length of the MERL file in bytes
3. **Code Length:** the length of the header plus the MIPS machine code

The MIPS Program

The program in MIPS machine code. This program is loaded into the RAM location 0x0c, which is the location immediately following the header.

Relocation and External Symbol Table

Contains relocation information and the external symbol definitions and external symbol reference (discussed later).