Lab 5

CMPUT 229

University of Alberta

Outline

Lab 5: The Main Idea

- You will implement a code translator.
- This translator will take MIPS code and translate it to equivalent ARM code that will **perform the same**.
- ARM is the acronymn foor *Advanced RISC Machine*.
  - As well as MIPS, it has a RISC architecture and a reduced number of registers.
  - Mostly used in embebbed electronics and have been used traditionally for mobile phones due to their reliable energy management.

Lab 5: The Main Idea

- Your translator must translate from *MIPS* binary to *ARM* binary.
- Most of the translation should be straightforward except for branches: in ARM, a MIPS branch or jump instruction might be translated to more than one instruction, thus invalidating the offset for most operations.
- There are several ways of dealing with this issue. You might choose whichever you like as long as it works. We propose a two-pass system:

1. Create two tables with one entry per MIPS line code: MIPS-to-ARM and Branch table. Each of these tables will use the same offset regardless of the base address for easy indexing.
2. In the **first pass**, translate to ARM without specifying the branch address.
   - In the MIPS-to-ARM table, write the address of the corresponding ARM.
   - In the Branch table, write the address of the corresponding branch target using the MIPS-to-ARM table if the instruction is a branch, 0 otherwise.

Lab 5: The Main Idea

- In the **second pass**, look for non-zeros on the branch table and update the ARM translated binary with the offset, easy calculated as a subtraction.
  1. Subtract (`target address - source adress - offset`) on the MIPS-to-ARM table.

Lab 5: The Main Idea

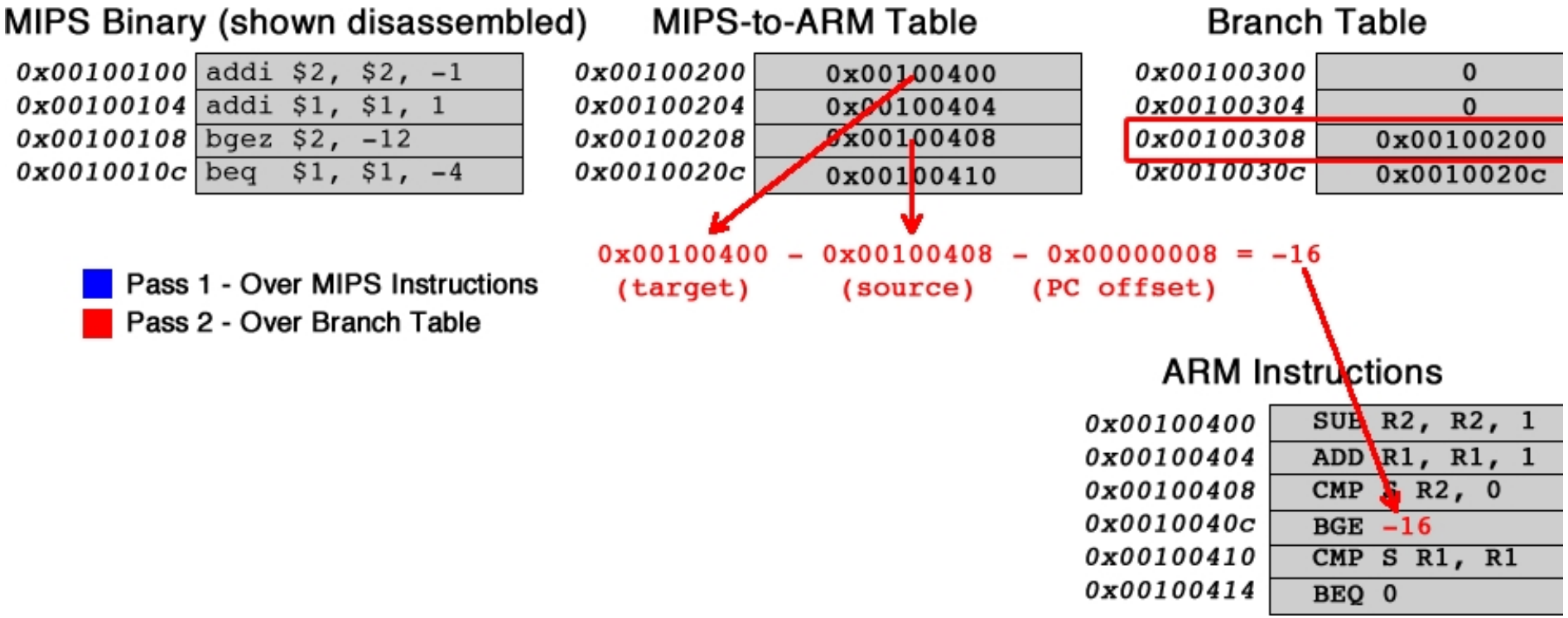

Figure: Illustrative Example

Lab 5: Tips

- We don't want all the possible instructions translated! Please refer to the specification for a table and more details.
- Look for the register translation table.
- You must create just one method `MIPStoARM` which does all the translation.
  - Args: $a0 = pointer to memory containing a MIPS function. (0xFFFFFFFF is a flag for EOF)
  - Return values: $v0 = number of ARM instructions generated, $v1 = a pointer to the first ARM instruction in memory.
- On the website, you can find test programs and data that might be useful.
- Please submit on time and read instructions carefully. We will adhere to specifications on the website for marking.

Lab 5 Questions?