

Lab 5

CMPUT 229

University of Alberta

Outline

1 Lab 5 Assignment

- Main Idea
- Tips
- Questions

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 acronym for *Advanced RISC Machine*.
 - As well as MIPS, it has a RISC architecture and a reduced number of registers.
 - Mostly used in embedded 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 address - offset) on the MIPS-to-ARM table.

Lab 5: The Main Idea

MIPS Binary (shown disassembled)

| | |
|------------|-------------------|
| 0x00100100 | addi \$2, \$2, -1 |
| 0x00100104 | addi \$1, \$1, 1 |
| 0x00100108 | bgez \$2, -12 |
| 0x0010010c | beq \$1, \$1, -4 |

MIPS-to-ARM Table

| | |
|------------|------------|
| 0x00100200 | 0x00100400 |
| 0x00100204 | 0x00100404 |
| 0x00100208 | 0x00100408 |
| 0x0010020c | 0x00100410 |

Branch Table

| | |
|------------|------------|
| 0x00100300 | 0 |
| 0x00100304 | 0 |
| 0x00100308 | 0x00100200 |
| 0x0010030c | 0x0010020c |

- Pass 1 - Over MIPS Instructions
- Pass 2 - Over Branch Table

$$0x00100400 - 0x00100408 - 0x00000008 = -16$$

(target) (source) (PC offset)

ARM Instructions

| | |
|------------|---------------|
| 0x00100400 | SUB R2, R2, 1 |
| 0x00100404 | ADD R1, R1, 1 |
| 0x00100408 | CMP R2, 0 |
| 0x0010040c | BGE -16 |
| 0x00100410 | CMP S R1, R1 |
| 0x00100414 | BEQ 0 |

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?