CS 251 — Lecture 1

Bartosz Antczak *Instructor: Stephen Mann* *January 3, 2017*

1.1 MIPS

MIPS (Microprocessor without Interlocked Pipelined Stages) is an assembly language. An assembly language is what high-level code (such as C++ or Java) gets converted into through a compiler. MIPS operates using simple operations (addition, subtraction, goto, conditional goto). These instructions operatre on two types of data — registers and RAM.

There are 32 registers (which can be interpreted essentially as variables). Each register can store 32 bits or four bytes of information. The notation for a register is \$n, which represents a variable that holds an integer value in the n-th register.

1.1.1 MIPS instructions

We have three types of MIPS instructions

- R-Format: adds the values stored in two registers and stores them in another register (e.g., `add $1, $2, $3` — adds \$2 and \$3 and stores in \$1)
- I-Format: similar to R-format, but adds an immediate value to one register and stores in another register (e.g., `addi $1, $2, 100` — adds the immediate value of 100 to \$2, stored in \$1)
- J-Format: used for branching, discussed later

These commands are stored in memory with an address that is always a multiple of 4.

Example 1.1.1. *MIPS instructions example (each instruction has their own distinct memory address)*

Memory	Instruction
100:	`add $1, $0, $0`
104:	`addi $2, $0, 6`
108:	`addi $1, $1, 5`
112:	`addi $2, $2, -1`

1.1.2 MIPS Control Flow

In MIPS, the only control flow we have is *jump* (unconditional goto) and *beq* (conditional goto)

- `jump`: jumps to the memory address that is equal to four times the immediate argument (e.g., `j 28` jumps to memory address $28 \times 4 = 112$)
- `beq`: compares two registers — if they're equal, add the memory address plus 4 times the constant argument to the PC counter; otherwise don't do anything (e.g., if \$1 and \$2 are equal, then `beq $1, $2, 100` will jump to memory address PC + $(4 \times 100) + 4$)
- `bne`: just like `beq` but it branches if the values are not equal

1.1.3 Memory access

More registers allow for faster compiling; however, clearly 32 registers are not enough to store data of most programs, thus we have special MIPS instructions to access 4GB of RAM. To work with this data, we use two commands:

- Load word (`lw $n, k($m)`): reads word from memory address k + \$m and stores in register \$n (e.g., `lw $1, 100($2)` reads the value stored at memory address 100+\$2 (M[100+\$2]) and stores the result in register \$1)
- Store word (`sw $n, k($m)`): takes value from register \$n and writes it to memory with address k + \$m (e.q., `sw $1, 100($2)` writes the value in register \$1 to M[100+\$2])

When selecting memory addresses to read and write from, it must be a multiple of 4.