Memory Access Operations

5 min

Memory access and the control of stored information is an incredibly powerful aspect of Assembly while also being one of the most confusing.

As we write Assembly, we control exactly where every piece of information is stored. This means it is on us to choose if a value is stored in a register for immediate use or in alternate memory to be loaded from later.

In the MIPS Assembly language, all calculations are performed exclusively on registers. Since we are limited on the number of registers we have, when we no longer need to use data right away we push it back into memory to access later.

The statements that provide these push and pull operations are LW and SW, which stand for LOAD WORD and STORE WORD.

A *word* is a fixed length of data defined by the ISA. In the MIPS32 architecture that we have been using, a word is a 32-bit value.

In this example we will load a number value from memory, add 15 to it, and save it back to memory:

1. LW loads the value stored in the memory address of Register 3 into Register 1

LW $1, ($3)

to Clipboard

1. ADDI adds the value of Register 1 and the constant, 15, and stores the result in Register 1

ADDI $1, $1, 15

to Clipboard

1. SW stores the value in Register 1 to the memory address of Register 3

SW $1, ($3)

to Clipboard

1. XOR is used to store zero in Register 1, essentially resetting the register.

XOR $1, $1, $1

to Clipboard

Unlike high-level languages, in Assembly, it is the programmer's responsibility to remember where pieces of data are stored and ensure not to overwrite it. Registers are limited and memory takes time to access, so it is a careful balancing act of keeping data available or pushing it deeper into memory.

Instructions

1. Checkpoint 1 Passed

1.

Create a new variable, answer_1, and set it equal to the opcode used to store a word value to a location.

Capitalize the whole answer.

Hint

The opcode is the symbol, usually two to four characters, used to reference the operation and not the whole name of the operation. For instance, the Assembly function “Load Word Right” has an opcode LWR.

2. Checkpoint 2 Passed

2.

Create another variable, answer_2, and set it equal to the opcode used to load a word value from memory.

Hint

The opcode is the symbol, usually two to four characters, used to reference the operation and not the whole name of the operation. For instance, the Assembly function “Load Word Right” has an opcode LWR.

memory_operations.py

```
# Write your answers here:

answer_1 = 'SW'

answer_2 = 'LW'
```