Memory Addressing: Direct and Indirect

6 min

You may have noticed the use of dollar signs and parentheses in our previous code. These are some MIPS language conventions used to denote direct and indirect memory referencing.

Let's assume that Register 5 has the value 1101000111_2 (839_{10}) stored in it. Let us also assume that there is a piece of memory with the address 839 that contains the value 1000111000111_2 (4551_{10}).

When we directly reference something in memory, we are telling the assembler that we want the value that is stored in that exact location.

Let's take a look at an Assembly code example that adds the value of Register 5 to Register 5 and saves the result in Register 6:

ADD $5, $5, $6

to Clipboard

Register 6 now contains 11010001110_2 (1678_{10}). 839 + 839 = 1678

When we indirectly reference memory, we are telling the assembler that we don't want to use the value of what is stored in the reference, we want to use that value as the address to another memory location and use the value stored there.

LW $4, ($5)
ADD $5, $4, $6

to Clipboard

In MIPS we can only perform operations on registers so the first step is to retrieve the value from memory with the Load Word statement and save it into Register 4. When we use the parentheses around Register 5, we are now telling the assembler to go and find the data that is stored in the memory address 839 which was 1000111000111_2 (4551_{10}).

At the conclusion of our ADD statement, Register 6 now contains the value 1010100001110_2 (5390_{10}). 4551 + 839 = 5390.

Instructions

1. Checkpoint 1 Passed

1.

Create a new variable, answer_1, and set it equal to the type of memory addressing that uses the value stored in one location as the address to reference another value.

When creating your answer, make sure the text is in all lowercase.

Hint

When we reference memory in this way, we are telling the CPU to go to the location contained within the reference and load a value from there.

2. Checkpoint 2 Passed

2.

Create another variable, answer_2, and set it equal to the symbols that surround an indirect memory reference in MIPS.

Hint

In most higher-level languages these symbols are reserved to pass in arguments to a function, in a roundabout sort of way, you can associate this concept to Assembly as well. We are passing a reference to a value stored in another location.

memory_addressing.py

```
# Your code below here:

answer_1 = 'indirect'

answer_2 = '()'
```