Instruction Formatting

3 min

In the previous exercise, we created the following OPCODE example:

000001 01001 10111 0001101001010110

Copy to Clipboard

We now know that the first part of the bit sequence is the OPCODE, but what about the rest? The rest of the message is the operands, memory locations, and additional functions that the

Preview: Docs Loading link description

processor

will perform.

We'll take the simple multiplication of 5 * 6 to visualize two different ISA styles, CISC and RISC, before we start to decipher the code beyond the OPCODE.

CISC

Preview: Docs Loading link description

machine code

is so long because the goal of CISC is to reduce the total number of instructions that are fed into the hardware. It may take multiple cycles of hardware to process the instruction, but it will still get done. The original purpose was to reduce the required memory for a program because memory was very expensive and consumed lots of space:

Machine Code:
00101001000100101010010101010010100010101010100111101001001001010010101000101001010
00101000011010001111011010110101011101001
Readable Code: MUL 5 * 6

Copy to Clipboard

RISC on the other hand, would take a CISC instruction and break it up into several very simple tasks that each require one cycle to complete. This may require more operations, but allows for multiple sequencing, or pipelining, of tasks to make up for it. Essentially, since all tasks take the same amount of time to complete, they can be executed more efficiently like so:

Machine Code: 00101001101001001001000100011100
Readable Code: LOAD 5 from REG 1

Machine Code: 00101001101001111001000100011100
Readable Code: LOAD 6 from REG 2

Machine Code: 00011000101001010101111000110001
Readable Code: MUL 5 * 6

Machine Code: 00111001101010100000111101010011

Readable Code: STORE 30 in REG 3

Copy to Clipboard

As we begin breaking up the remaining bits in our machine code, we will use a specific type of RISC instruction architecture called MIPS, or *Microprocessor without Interlocked Pipeline Stages*. It has a fixed 32-bit instruction length, limited instructions, and is used by most post-secondary educators for its ease of understanding. Here is a link to the documentation if you want to learn more about the MIPS architecture.

Instructions

Take a look at some of the MIPS instructions in the console. When you are finished, what characteristics make this a great example of a RISC ISA?

Possible Answers

- Fixed length
- Limited number of possible instructions
- Requires only simple hardware to process
- Small instruction set require less power and hardware to process

Concept Review

Want to quickly review some of the concepts you've been learning? Take a look at this material's cheatsheet!

Community Support

Still have questions? Get help from the **Codecademy community**.

Code Editor

script.py

Fullscreen Code Editor

1

2

3

4

5

6

7

8

9

10

11

12

13

14

15

16

17

18

19

MIPS_instruction1 = '10101101111010001000000000000000'

STORE WORD:

- stores a word to memory

MIPS_instruction2 = '00010111011000100000010000011100'

BRANCH ON NOT EQUAL

- compares General Purpose Registers (GPRs) then do a conditional branch

MIPS_instruction3 = '00011111001000010010010000000010'

BRANCH ON GREATER THAN ZERO

- tests a GPR then does a conditional branch

MIPS_instruction4 = '01101000100000100000011100010000'

LOAD DOUBLE WORD LEFT

- loads the most-significant part of a doubleword from an unaligned memory address

MIPS_instruction5 = '00101001000100001100100100100000'

SET ON LESS THAN IMMEDIATE

- records the result of a less-than comparison with a constant

ctrl + enter

Run

7/10

Back (alt + ,)

Back

Next (alt + .)

Next

MIPS Instructions | Instruction Set Architecture | Codecademy