ISA Instructions

4 registers:

|  |  |  |  |
| --- | --- | --- | --- |
| 00 | 01 | 10 | 11 |
| $0 | $1 | $2 | $3 |

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Instr** | **OP**  **code** | **Details** | | | | | **Example** |
| *Syntax* | *Machine code* | *Supported registers* | *Supported imm. range* | *General behavior* |
| refresh | 0000 | refresh rt imm. | 0000 ttii | [0, 1, 2, 3] | [-2, 1] | Initializes register rt into the given immediate.  rt = imm. | **refresh** $0, 0 |
|
| addi | 0001 | addi rt imm. | 0001 ttii | [0, 1, 2, 3] | [-2, 1] | Adds register rt with the given immediate and stores result to rt.  rt = rt + imm. | **addi** $0, 1 |
| addu | 0010 | addu rt, rs | 0010 ttss |  | X | Adds registers rt and rs together and stores the unsigned result to rt.  rt = rt + rs | refresh $0, 0  refresh $1, 1  **addu** $1, $0 |
| store | 0011 | store rt, rm | 0011 ttmm |  | X | Stores the value of register rt into the memory address.  M[rm] = rt | refresh $2, 0  **store** $2, $2 |
| mult | 0100 | mult rt, rs | 0100 ttss |  | X | Multiplies rt and rs into a 16 bit number then stores the 8 MSB into $1 and the 8 LSB into $0.  temp16 = rt x rs  $0 = temp16 & 0x00FF  $1 = temp16 & 0xFF00 | **mult** $1, $1 |
| splice | 0101 | splice $1, $0 | 0101 0100 | [0, 1] | X | Drops the 4 LSB of $1 and 4 MSB of $0 then merge into $1.  $1 = $1 & 0x0F  $0 = $0 & 0xF0  $1 = $1 + $0 | **splice** $1, $0 |
| bne | 0110 | bne rt, rs | 0110 ttss |  | X | Branch if register rt is not equal to 0 otherwise don’t.  If rt != 0,  pc+=rs  else  pc++ | addi $0, -2  addi $0, -2  **bne** $2, $0 |

<https://github.com/lohe987/ECE366Spring2019_Sample/tree/master/Python>

Personal log:

3/28/2019 – UIC SPH 11th floor, 10 AM – 2 PM: finished my register recycling approach, redesigned referenced prpg.asm to use only 4 registers, created a new ISA architecture using only 4 registers

3/31/2019 – Apartment, 10 AM – 10 PM: finished and debugged the python simulator for our ISA.