ISA Instructions

4 registers:

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

Special registers (*rx)*:

|  |  |
| --- | --- |
| 0 | 1 |
| *uno* | *dos* |

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Instr.** | **OP**  **code** | **Details** | | | | | **Example** |
| *Syntax* | *Machine code* | *Supported registers* | *Supported imm. range* | *General behavior* |
| refresh  (0) | 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  (1) | 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  (2) | 0010 | addu rt, rs | 0010 ttss | [0, 1, 2, 3] | 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  (3) | 0011 | store rt, rs | 0011 ttss | [0, 1, 2, 3] | X | Stores the value of register rt into the memory address.  M[rs] = rt | refresh $2, 0  **store** $2, $2 |
| mult  (4) | 0100 | mult rt, rs | 0100 ttss | [0, 1, 2, 3] | 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  (5) | 0101 | splice $1, $0 | 0101 0100 | [0, 1] | X | Drops the 4 LSB of $1 and 4 MSB of $0 then merge into $1.  $0 = $0 & 0x0F  $1 = $1 & 0xF0  $1 = $1 + $0 | **splice** $1, $0 |
| jne  (6) | 0110 | jne rt, imm., rx | 0110 ttix | [0, 1, 2, 3,]  Special registers:  *uno*  *dos* | [0, 1] | If register rt is not equal to the given imm., jump to the value of register *rx.*  If rt != imm.,    pc = rx  else    pc++ | addi $2, -2  **jne** $2, 0, 1 |
| mark  (7) | 0111 | mark rx | 0111 XXXx  \*X - don’t cares | Special registers:  *uno*  *dos* | X | Stores the next pc value into register rx.  rx = pc + 1 | **mark** 1 |
| slti  (8) | 1000 | slti rt, imm. | 1000 ttii | [0, 1, 2, 3] | [0, 3] | If register rt value is less than the given immediate, set value of register 4 to 1, else set to 0.  rt value < 0?,      $3 = 1:0 | **slti** $1, 0 |
| beq  (9) | 1001 | beq rt, imm. | 1001 ttii | [0, 1, 2, 3] | [-2, 1] | If register rt value is equal to 0, update pc with given immediate.  if rt == 0,    pc = pc + imm.  else    pc++ | **slti** $1, 0  **beq** $1, 1 |
| sll  (A) | 1010 | sll rt, imm. | 1010 ttii | [0, 1, 2, 3] | [0, 3] | Logic shift register rt to the left by the given immediate and store it back to rt.  rt = rt << imm. | **sll** $1, 1 |
| srl  (B) | 1011 | srl rt, rs | 1011 ttss | [0, 1, 2, 3] | X | Logic shift register rt to the right by the value of register rs and store it back to rt.  rt = rt >> rs | **srl** $3, $1 |
| load  (C) | 1100 | load rt, rs | 1100 ttss | [0, 1, 2, 3] | X | Loads the value of memory address into the register rt.  rt = M[rs] | refresh $2, 0  **load** $2, $2 |

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 using only 4 registers

3/31/2019 – Apartment, 10 AM – 10 PM: adapted our own ISA syntax using the PRPG assembly file, finished and debugged the python simulator for part A and B.

4/1/2019 – UIC library 1st floor, 6 PM – 10 PM: debugged python simulator, performed all seed inputs and successfully displayed their correct outputs