**Assignment 1:**

|  |  |  |
| --- | --- | --- |
|  | Run the first instruction: | Run the second instruction: |
| $s0 | 0x00003007 | 0x00000000 |
| $pc | 0x00400004 | 0x00400008 |

* Inititally, In Text Segment,

+ The machine code of addi : 0x20103007.

Convert to binary : 0010 0000 0001 0000 0011 0000 0000 0111

I-type : 001000 00000 10000 0011000000000111

* Opcode :8 =>addi
* rs : 00000 => $0 => $zero
* rt : 10000 => $16 =>$s0
* immediate : 0011000000000111 => 0x3007

+ The machine code of add : 0x00008020.

Convert to binary : 0000 0000 0000 0000 1000 0000 0010 0000

R-type : 000000 00000 00000 10000 00000 100000

* Opcode :0
* rs : 00000 => $0 => $zero
* rt : 00000 => $0 =>$zero
* rd: 10000 => $16 => $s0
* sh : 00000 => 0
* function: 100000 => 32 => add
* When we edit the code to : addi $s0, $zero, 0x2110003d :
* 0x2110003d is a number of 32 bit but the RAM just has 16 bit
* Assign twice by two instructions lui and ori.

the instruction is not in instruction set. It is assembled by 3 instruction lui, ori and add.

**Assignment 2:**

|  |  |  |
| --- | --- | --- |
|  | Run the first instruction: | Run the second instruction: |
| $s0 | 0x21100000 | 0x2110003d |
| $pc | 0x00400004 | 0x00400008 |

* After each instruction, $pc increases 4 bit.
* The first bytes in the command area **match** the Code column in the Text Segment window.

**Assignment 3:**

* In Text Segment, the machine code and the command format are not the same .

The instruction li is replaced by lui , ori and addiu.

Li is a pseudo instruction, so when compiled it was compiled back to Basic instruction.

* We must assign a 32-bit number twice each 16 bits => Need lui and ori.
* When we assign a 16-bit number or less, just need only one assignment instruction.

**Assignment 4**:

The change of register :

|  |  |  |
| --- | --- | --- |
| Instruction | $pc |  |
| 1 | 0x00400004 | $t1 : 0x00000005 |
| 2 | 0x00400008 | $t2 : 0xffffffff |
| 3 | 0x0040000c | $s0 : 0x0000000a |
| 4 | 0x00400010 | $s0 : 0x00000009 |

After finishing the program, the result is true.

+ The machine code of the 1st addi : 0x20090005.

Convert to binary : 0010 0000 0000 1001 0000 0000 0000 0101

I-type : 001000 00000 01001 0000000000000101

* Opcode :8 =>addi
* rs : 00000 => $0 => $zero
* rt : 01001 => $9 =>$t1
* immediate : 0000000000000101=> 0x5

+ The machine code of the 2st addi : 0x200affff.

Convert to binary : 0010 0000 0000 1010 1111 1111 1111 1111

I-type : 001000 00000 01010 1111111111111111

* Opcode :8 =>addi
* rs : 00000 => $0 => $zero
* rt : 01010 => $10 =>$t2
* immediate : 0xffff

=>>Matching the instruction format I. The machine code and assembly language are similar at: The last 5 bits of machine code are input to the assembly language.

+ The machine code of the 1st add : 0x01298020.

Convert to binary : 0000 0001 0010 1001 1000 0000 0010 0000

R-type : 000000 01001 01001 10000 00000 100000

* Opcode :0
* rs : 01001 => $9 => $t1
* rt : 01001 => $9 =>$t1
* rd: 10000 => $16 => $s0
* sh : 00000 => 0
* function: 100000 => 32 => add

+ The machine code of the 2nd add : 0x020a8020.

Convert to binary : 0000 0010 0000 1010 1000 0000 0010 0000

R-type : 000000 10000 01010 10000 00000 100000

* Opcode :0
* rs : 10000 => $16 => $s0
* rt : 01010=> $10 =>$t2
* rd: 10000 => $16 => $s0
* sh : 00000 => 0
* function: 100000 => 32 => add

=>>Matching the instruction format R.

**Assignment 5:**

**X multiplied by Y**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Instruction | $pc |  | $hi | $lo |
| 1 | 0x00400004 | $t1 : 0x00000004 | 0x00000000 | 0x00000000 |
| 2 | 0x00400008 | $t2: 0x00000005  $gp:0x10008000 | 0x00000000 | 0x00000000 |
| 3 | 0x0040000c | $s0 : 0x00000014 | 0x00000000 | 0x00000014 |
| 4 | 0x00400010 | $at : 0x00000003 | 0x00000000 | 0x00000014 |
| 5 | 0x00400014 | $s0:0x0000003c | 0x00000000 | 0x0000003c |
| 6 | 0x00400018 | $s1:0x0000003c | 0x00000000 | 0x0000003c |

When compiling, the instruction mul in line 8 is divided into two subinstruction,

The register $hi does not change.

The final result = 0x3c = 60. And the multiplication 3\*4\*5 =60

* The result is true

**Assignment 6:**

* The instruction la is divided into two instruction lui and ori.

The first 4 bytes of address of X,Y,Z is 4 bytes of lui.

The last 4 bytes of address of X,Y,Z is 4 bytes of ori

-The change of registers:

|  |  |  |
| --- | --- | --- |
| Instruction |  |  |
| la at line 8 | $at :0x10010000 | $t8 : 0x10010000 |
| la at line 9 | $at:0x10010000 | $t9: 0x10010004 |
| lw at line 10 | $t1:0x00000005 |  |
| Lw at line 11 | $t2:0xffffffff |  |
| Add at line 13 | $s0:0x0000000a |  |
| La at line 16 | $s0: 0x00000009 |  |
| Sw at line 17: | $t7:0x10010008 |  |

• lw instruction (load word): set this register with another register value (word data type)

• Command sw (store word): stores content from memory (word data type)

• The lb and sb instructions run similarly to the lw and sw commands, but with different data types (bytes instead of words).