Jiaxing Wu

Professor Michael Choi

CS350

Lab 4

Step 2:

|  |  |  |  |
| --- | --- | --- | --- |
| **Address** | **Pseudo instruction** | **Native instructions** | **Effect** |
| 00400028 | *li $t0, 3* | ori $8, $0, 3 | Add zero and immediate 3 and put that value in register 8 (the effect is like storing the constant 3 in register 8 because adding zero to the number doesn’t do anything to it) |
| 0040002c | *move $t1, $t0* | addu $9, $0, $8 | Add the unsigned number in register 8 to zero and store it in register 9(the effect is like moving it to register 9 because it only uses the target register → adding zero will not do anything to it) |
| 00400030 &  00400034 | *la $t2, var2* | lui $1, 4096 [var2]  ori $10, $1, 4 [var2] | The first native instruction loads the upper bound or the address in var2 (4096)10 or (1000)16 and store it in register 1. This part is split into two parts because the address, itself, is a 32 bit thing and a MIPS instruction is only 32 bits. Instead of using the R-format, we use the I-Format so we can store the upper bound of the address first and then the lower bound. For the second part, it adds the lower bound (4)10 or (0004)16 with the upper bound in register 1 and store that into register 10 |
| 00400038 &  0040003c | *lw $t3, var2* | lui $1, 4096  lw $11, 4($1) | Load constant the (4096)10 to register 1 in hex for the upper bound. Then, load word from the base address of register 1 with offset of 4 to register 11. |
| 00400040 &  00400044 | *sw $t2, var1* | lui $1, 4096  sw $10, 0($1) | Load constant the (4096)10 to register 1 in hex for the upper bound. Then, store word from register 10 (currently holds var2) to the base address in register 1 with offset of 0 in the memory |

Step 3:

|  |  |
| --- | --- |
| **Decimal constant** | **16 bit hexadecimal** |
| 4096 | 1000(for the upper bound)0000(for the lower bound) |

Step 4:

|  |  |  |
| --- | --- | --- |
| **Register** | **Before executing lui** | **After executing lui** |
| $1 or $at | 0 | 10000000 |

Question 1:

ori $8, $0, 2882338816 should be executed

Step 5:

$t2 = 10000004

Question 2:

It first loads the upper bound of var2 which is the constant 4096 to register at. Then, it adds that to the lower bound of var2 which is the constant 4 and saves the hex version in register t2

Question 3:

ori $8, $0, 2882339055 should be executed