Anthony Redamonti

Syracuse university

Homework 2 Question 2: Custom ISA

CIS-655 Advanced computer architecture

prof. mohammed abdallah

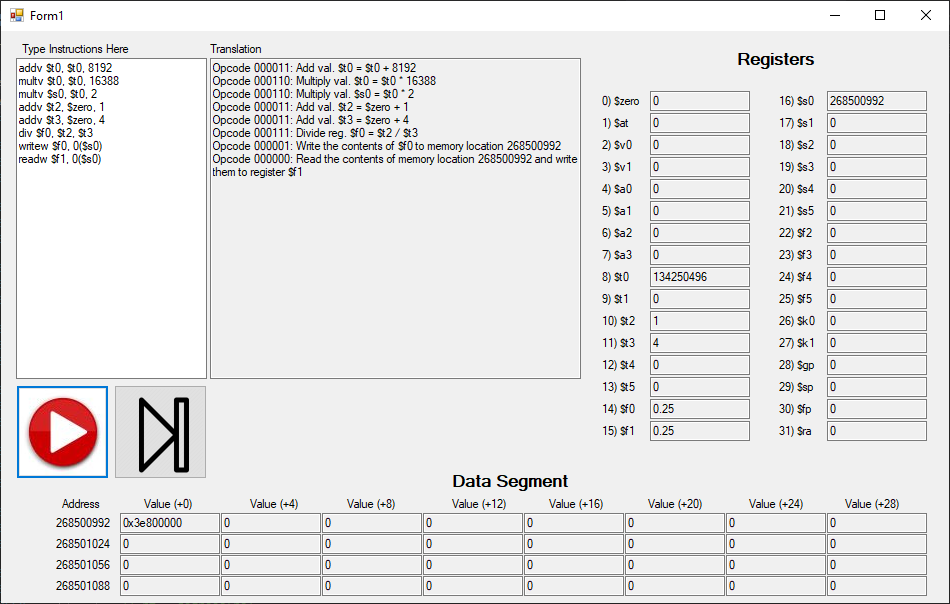
10/16/2021

The following program is written in C#. It is a code editor for a custom instruction set architecture detailed in this document. Below is a table of the opcodes. There are 50 in total. Do to the file size, the code is attached in a zip folder.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Opcode** | **Name** | **Action** | **Opcode Bitfields** | | | | |
| readw | Read word | Rt=\*(\*int)(offset + Rs) | 000000 | Rs | Rt | offset | |
| writew | Write word | \*(\*int)(offset + Rs) = Rt | 000001 | Rs | Rt | offset | |
| add | Add reg | Rt = Rs + Rd | 000010 | Rs | Rt | Rd | 0x00 |
| addv | Add val | Rt = Rs + (signed val) | 000011 | Rs | Rt | signed 16-bit value | |
| sub | Subtract reg | Rt = Rs – Rd | 000100 | Rs | Rt | Rd | 0x00 |
| mult | Multiply reg | Rt = Rs \* Rd | 000101 | Rs | Rt | Rd | 0x00 |
| multv | Multiply val | Rt = Rs \* (signed val) | 000110 | Rs | Rt | signed 16-bit value | |
| div | Divide reg | Rt = Rs/Rd | 000111 | Rs | Rt | Rd | 0x00 |
| divrv | Divide reg val | Rt = Rs/val | 001000 | Rs | Rt | signed 16-bit value | |
| divvr | Divide val reg | Rt = val/Rs | 001001 | Rs | Rt | signed 16-bit value | |
| and | And reg | Rt = Rs & Rd | 001010 | Rs | Rt | Rd | 0x00 |
| or | Or reg | Rt = Rs|Rd | 001011 | Rs | Rt | Rd | 0x00 |
| xor | Xor reg | Rt = Rs^Rd | 001100 | Rs | Rt | Rd | 0x00 |
| nand | Nand reg | Rt = ~(Rs & Rd) | 001101 | Rs | Rt | Rd | 0x00 |
| nor | Nor reg | Rt = ~(Rs|Rd) | 001110 | Rs | Rt | Rd | 0x00 |
| not | Not reg | Rt = ~(Rs) | 001111 | Rs | Rt | 0x0000 | |
| andv | And val | Rt = Rs & val | 010000 | Rs | Rt | signed 16-bit value | |
| orv | Or val | Rt = Rs|val | 010001 | Rs | Rt | signed 16-bit value | |
| xorv | Xor val | Rt = Rs^val | 010010 | Rs | Rt | signed 16-bit value | |
| nandv | Nand val | Rt = ~(Rs & val) | 010011 | Rs | Rt | signed 16-bit value | |
| setlt | Set less than | Rt = 1 if Rs < Rd. Else 0. | 010100 | Rs | Rt | Rd | 0x00 |
| setltv | Set less than val | Rt = 1 if Rs < val. Else 0. | 010101 | Rs | Rt | signed 16-bit value | |
| atan2 | Arctan2 | Rt = arctan2(Rs/Rd) | 010110 | Rs | Rt | Rd | 0x00 |
| rand | Random | Rt = Rand(Rs, Rd) | 010111 | Rs | Rt | Rd | 0x00 |
| log | Logarithmic |  | 011000 | Rs | Rt | Rd | 0x00 |
| pow | Power |  | 011001 | Rs | Rt | Rd | 0x00 |
| hypot | Hypotenuse |  | 011010 | Rs | Rt | Rd | 0x00 |
| shiftr | Shift Right | Rt = Rs >> value | 011011 | Rs | Rt | unsigned 16-bit value | |
| shiftl | Shift Left | Rt = Rs << value | 011100 | Rs | Rt | unsigned 16-bit value | |
| sin | Sine | Rt = sin(Rs) | 011101 | Rs | Rt | 0x00 | 0x00 |
| cos | Cosine | Rt = cos(Rs) | 011101 | Rs | Rt | 0x00 | 0x01 |
| tan | Tangent | Rt = tan(Rs) | 011101 | Rs | Rt | 0x00 | 0x02 |
| csc | Cosecant | Rt = csc(Rs) | 011101 | Rs | Rt | 0x00 | 0x03 |
| sec | Secant | Rt = sec(Rs) | 011101 | Rs | Rt | 0x00 | 0x04 |
| cot | Cotangent | Rt = cot(Rs) | 011101 | Rs | Rt | 0x00 | 0x05 |
| sinh | Hyperbolic Sine | Rt = sinh(Rs) | 011101 | Rs | Rt | 0x00 | 0x06 |
| cosh | Hyperb. Cosine | Rt = cosh(Rs) | 011101 | Rs | Rt | 0x00 | 0x07 |
| tanh | Hyperb. Tangent | Rt = tanh(Rs) | 011101 | Rs | Rt | 0x00 | 0x08 |
| asin | Arcsine | Rt = arcsin(Rs) | 011101 | Rs | Rt | 0x00 | 0x09 |
| acos | Arccosine | Rt = arccos(Rs) | 011101 | Rs | Rt | 0x00 | 0x0a |
| atan | Arctan | Rt = arctan(Rs) | 011101 | Rs | Rt | 0x00 | 0x0b |
| sqrt | Square Root |  | 011101 | Rs | Rt | 0x00 | 0x0c |
| flr | Floor |  | 011101 | Rs | Rt | 0x00 | 0x0d |
| ceil | Ceiling |  | 011101 | Rs | Rt | 0x00 | 0x0e |
| trunc | Truncate | Rt = trunc(Rs) | 011101 | Rs | Rt | 0x00 | 0x0f |
| rnd | Round |  | 011101 | Rs | Rt | 0x00 | 0x10 |
| abs | Absolute Value | Rt = abs(Rs) | 011101 | Rs | Rt | 0x00 | 0x11 |
| deg | Degrees | Rt = Rs \* (180/) | 011101 | Rs | Rt | 0x00 | 0x12 |
| rad | Radians | Rt = Rs \* (/180) | 011101 | Rs | Rt | 0x00 | 0x13 |
| fact | Factorial | Rt = Rs! | 011101 | Rs | Rt | 0x00 | 0x14 |

The GUI features 32 general purpose registers. Included in these registers are 6 floating-point registers, $f0-$f5. They have the ability to store decimal numbers.

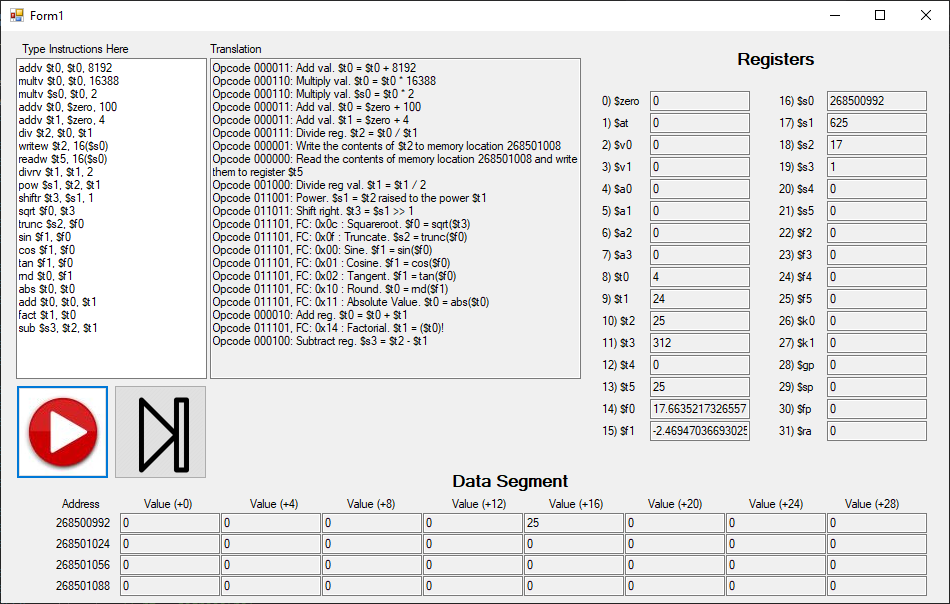
Example sequence 1:



Notice that the floating-point registers in the GUI display the decimal value. Normally these registers would store the decimal number in floating-point format. However, for readability, it was decided to display them in decimal format. If the user wants to see their floating-point representation, simply write the value to memory, and the program will convert the value for the user. If the user reads that value back from memory, the program will convert it back to decimal form. See the above example, Sequence 1.

Example sequence 2:

The example below displays the result from the following sequence of commands. Notice that because there are floating-point registers, the opcodes for trigonometry are more practical.



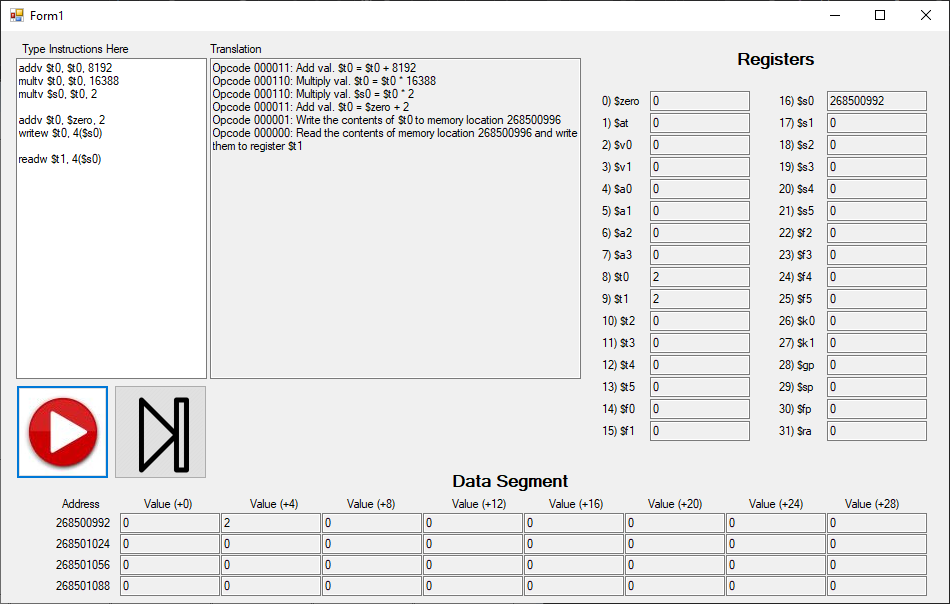
As the user executes each command, the program will print out the opcode and the registers involved in the instruction. The register values and the values in memory update as the user interacts with the GUI.

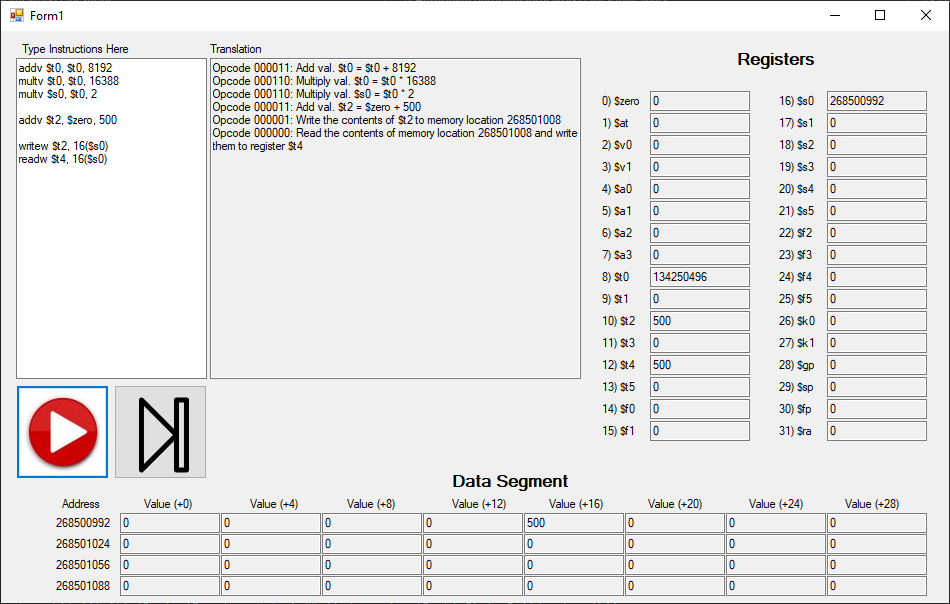
The user can either run the program using the play button or step through each instruction using the single step button shown above.

Readw : Opcode = 000000

Read a word of data from memory and store it in the register, Rt. The memory address is calculated as “Rs + address offset”.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Bits** | 26-31 | 21-25 | 16-20 | 0-15 |
| **Meaning** | Opcode | Rt | Rt | Address Offset |

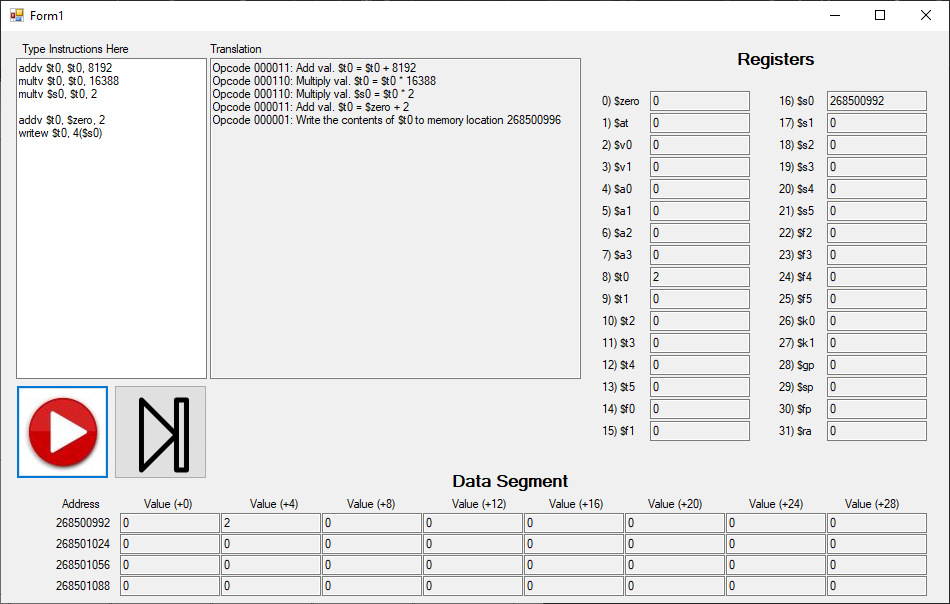


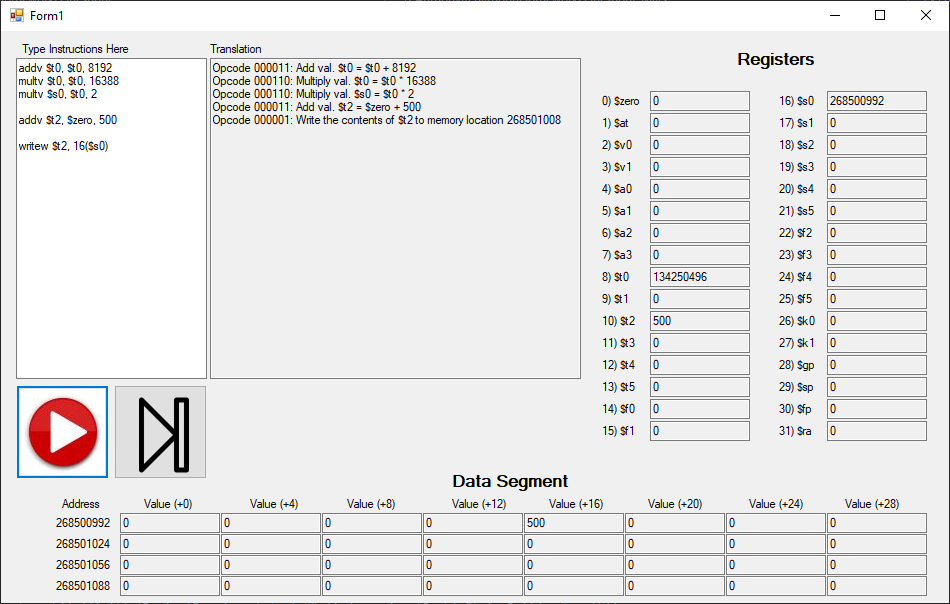


Writew: Opcode = 000001

Write a word of data stored in Rt to a memory address. The memory address is calculated as “Rs + address offset”.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Bits** | 26-31 | 21-25 | 16-20 | 0-15 |
| **Meaning** | Opcode | Rs | Rt | Address Offset |

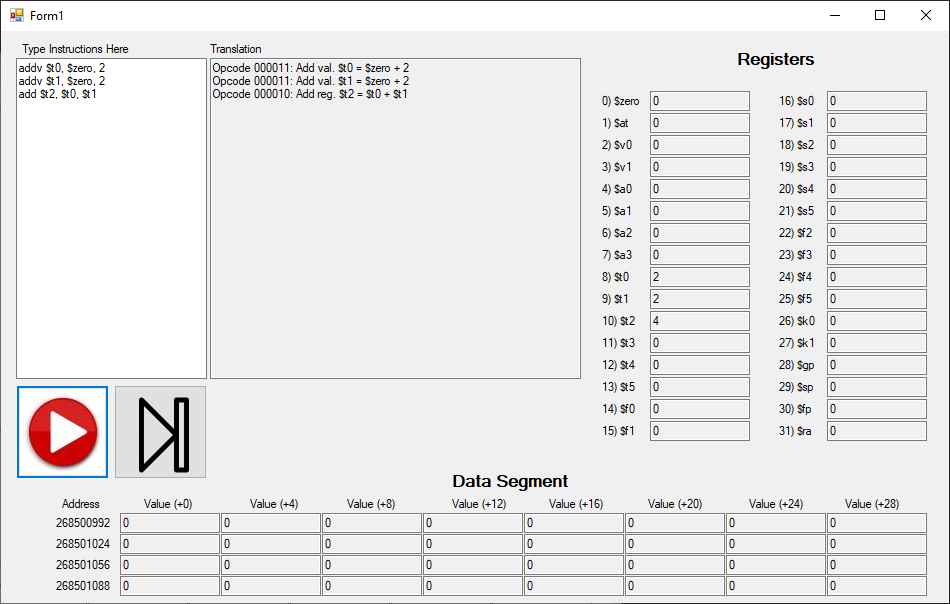


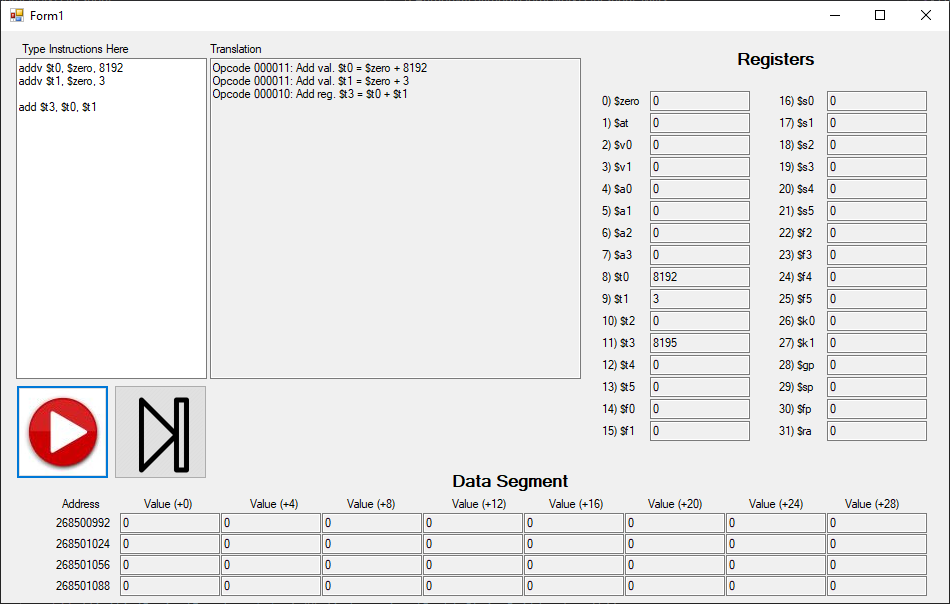


Add: Opcode = 000010

Add two registers together.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Bits** | 26-31 | 21-25 | 16-20 | 11-15 | 0-10 |
| **Meaning** | Opcode | Rs | Rt | Rd | 00000000000 |

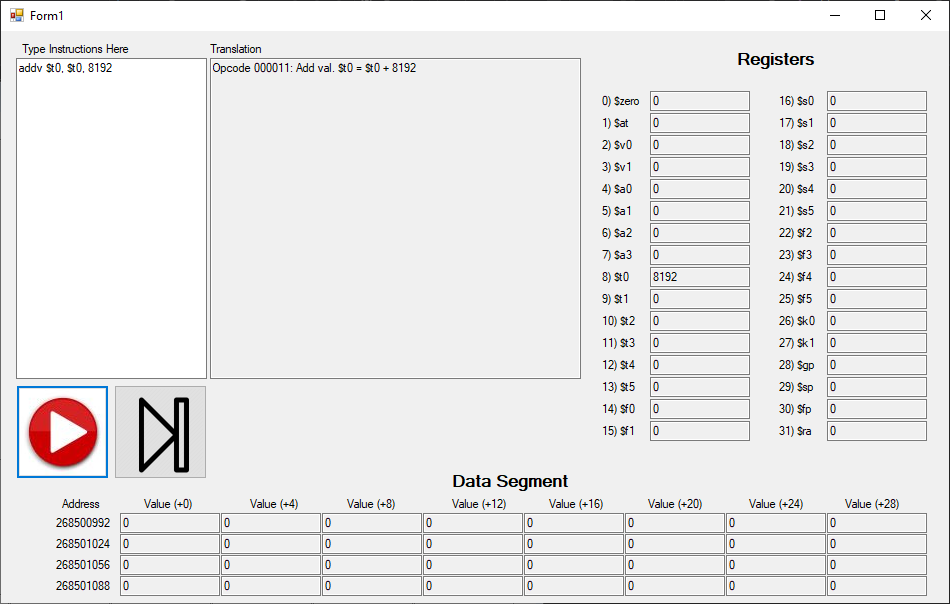


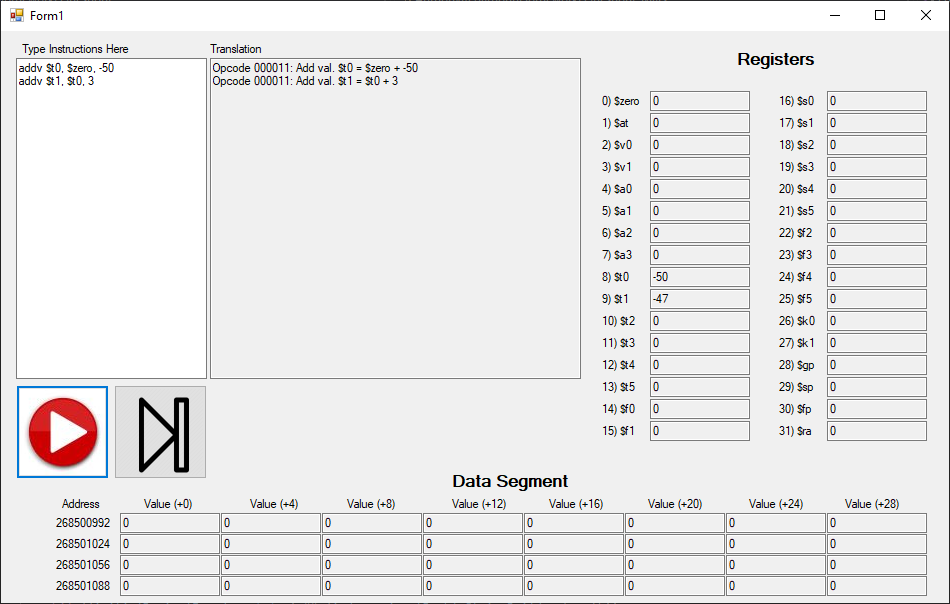


Addv: Opcode = 000011

Add a signed integer value to a register.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Bits** | 26-31 | 21-25 | 16-20 | 0-15 |
| **Meaning** | Opcode | Rs | Rt | Signed value |

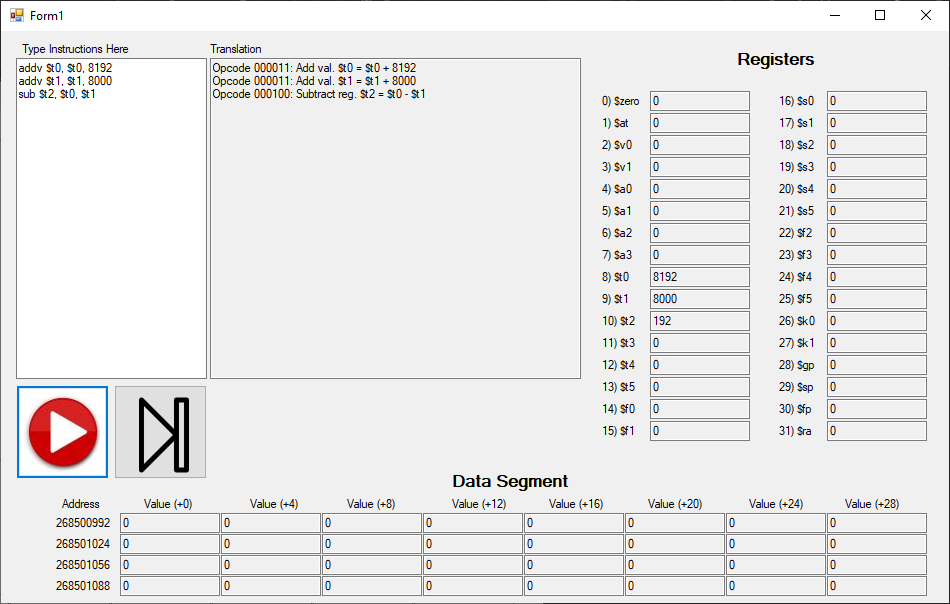


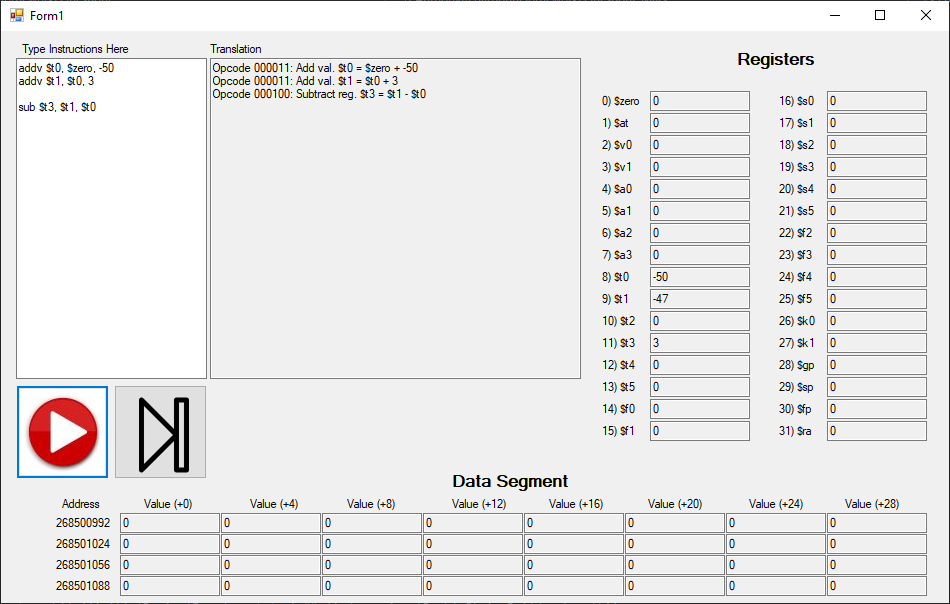


Sub: Opcode = 000100

Subtract two registers.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Bits** | 26-31 | 21-25 | 16-20 | 11-15 | 0-10 |
| **Meaning** | Opcode | Rs | Rt | Rd | 00000000000 |

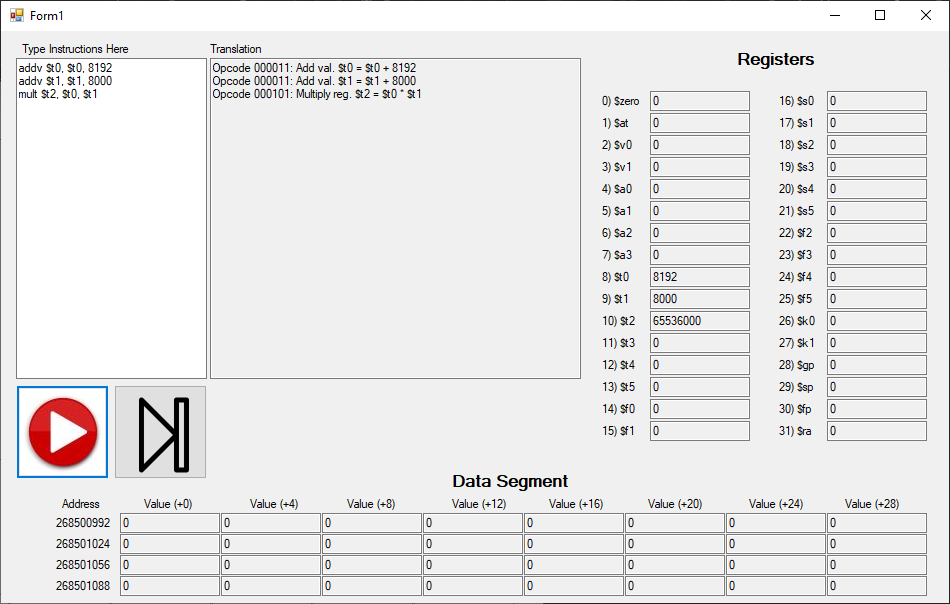


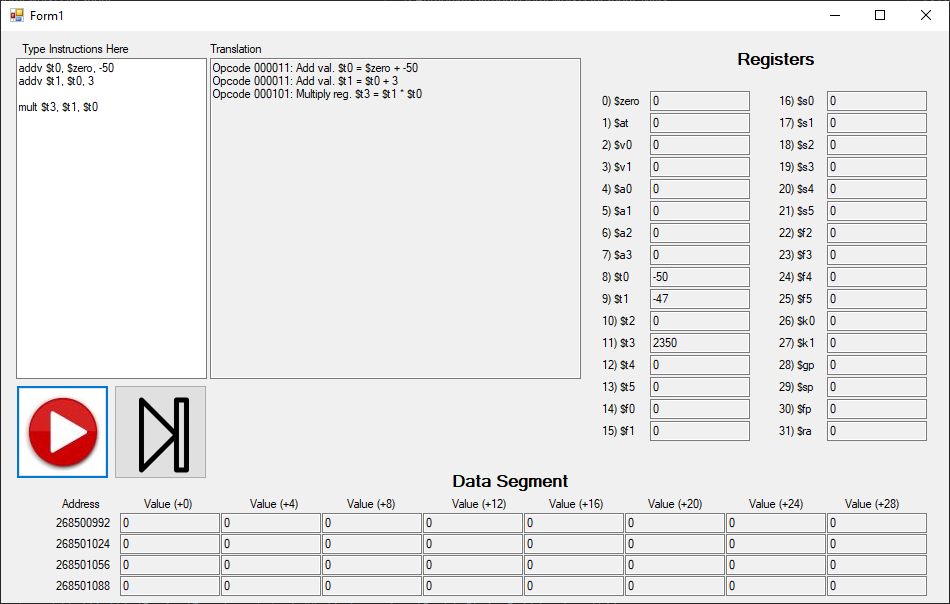


Mult: Opcode = 000101

Multiply two registers.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Bits** | 26-31 | 21-25 | 16-20 | 11-15 | 0-10 |
| **Meaning** | Opcode | Rs | Rt | Rd | 00000000000 |

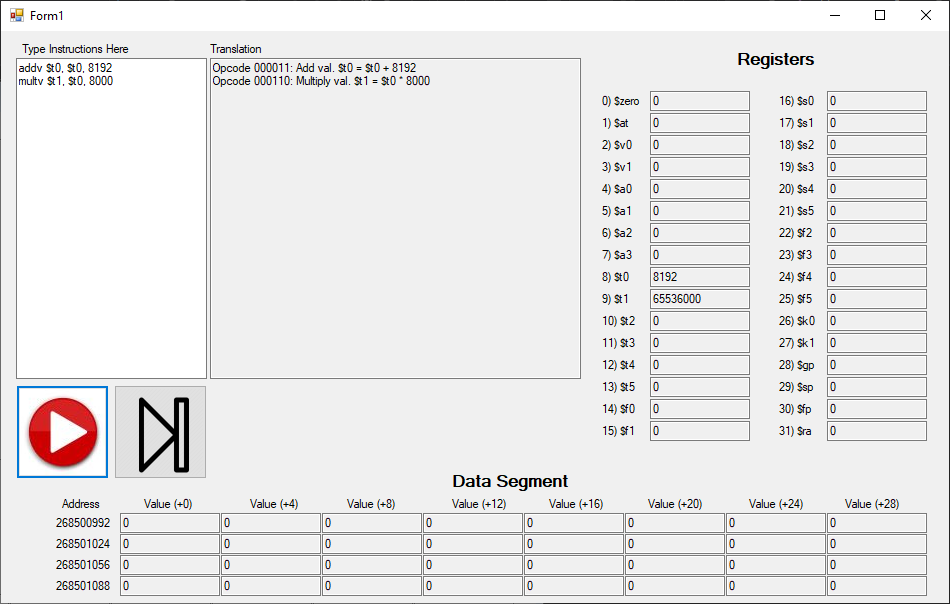


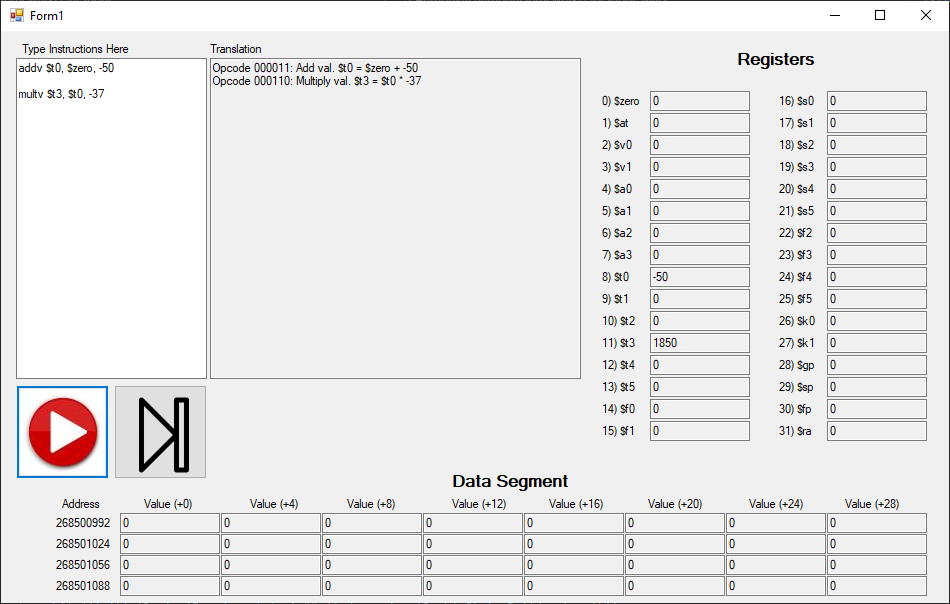


Multv: Opcode = 000110

Multiply a register by a signed 16-bit value.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Bits** | 26-31 | 21-25 | 16-20 | 0-15 |
| **Meaning** | Opcode | Rs | Rt | Signed value |

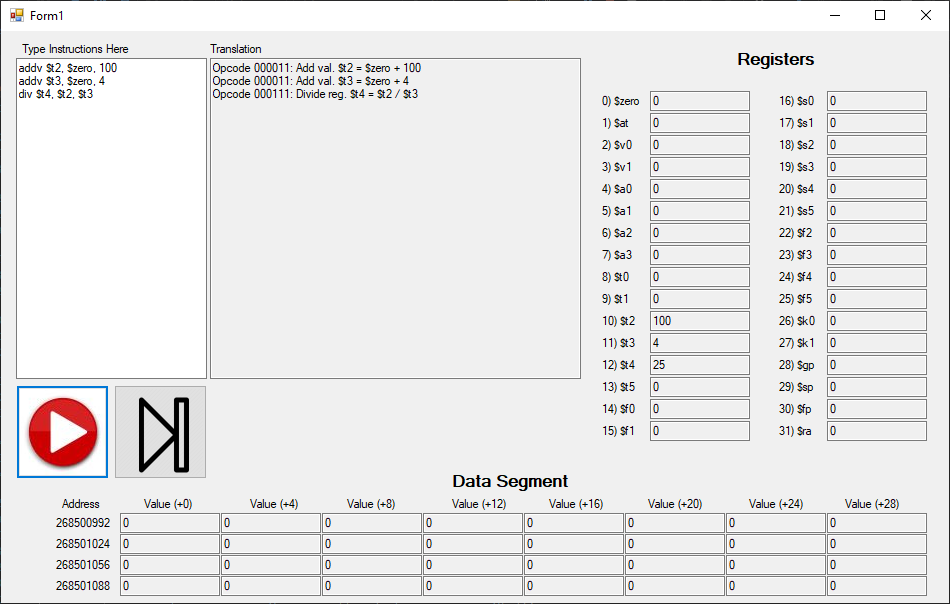


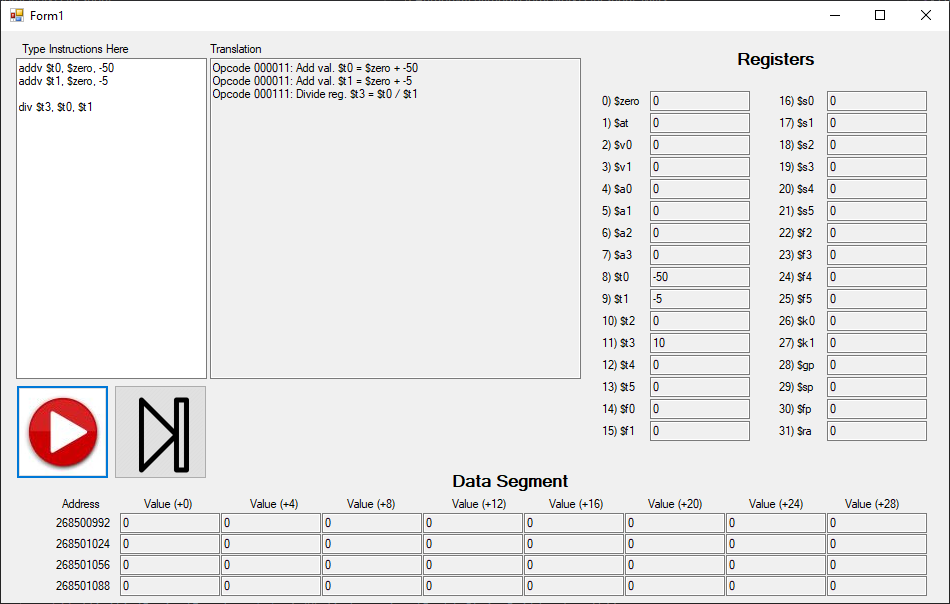


Div: Opcode = 000111

Divide two registers. Compatible with floating point and non-floating-point registers.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Bits** | 26-31 | 21-25 | 16-20 | 11-15 | 0-10 |
| **Meaning** | Opcode | Rs | Rt | Rd | 00000000000 |

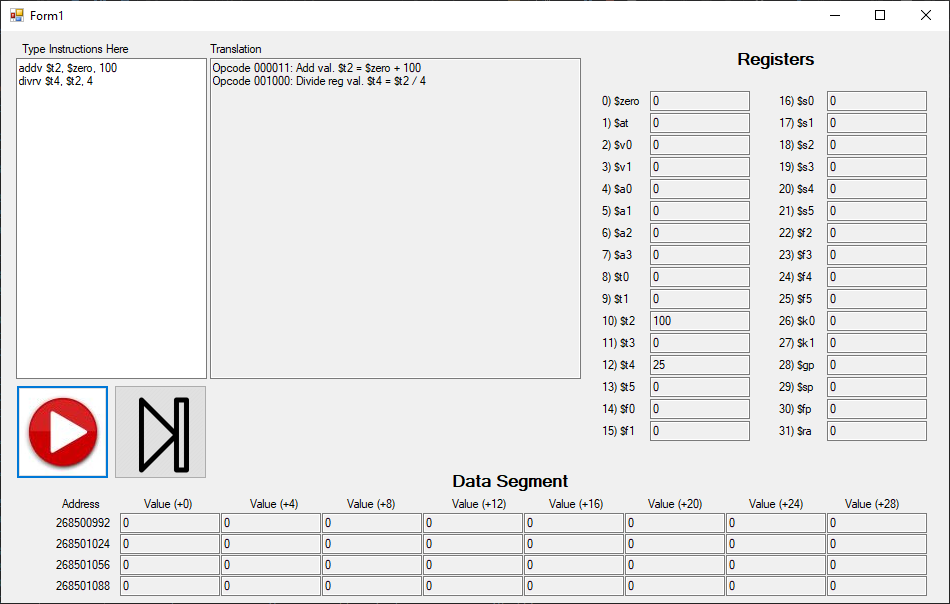


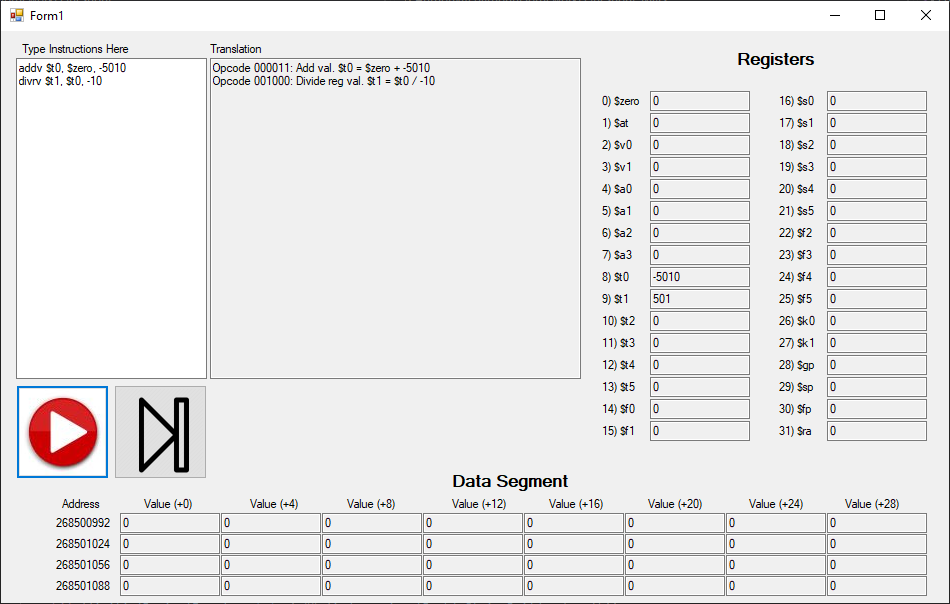


Divrv: Opcode = 001000

Divide a register by a 16-bit value. Compatible with floating point and non-floating-point registers.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Bits** | 26-31 | 21-25 | 16-20 | 0-15 |
| **Meaning** | Opcode | Rs | Rt | Signed value |

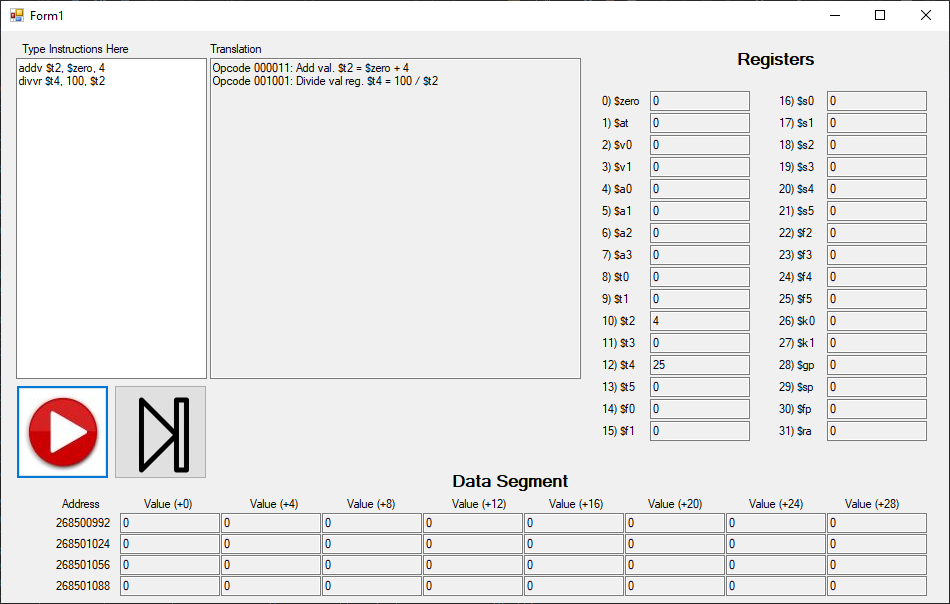


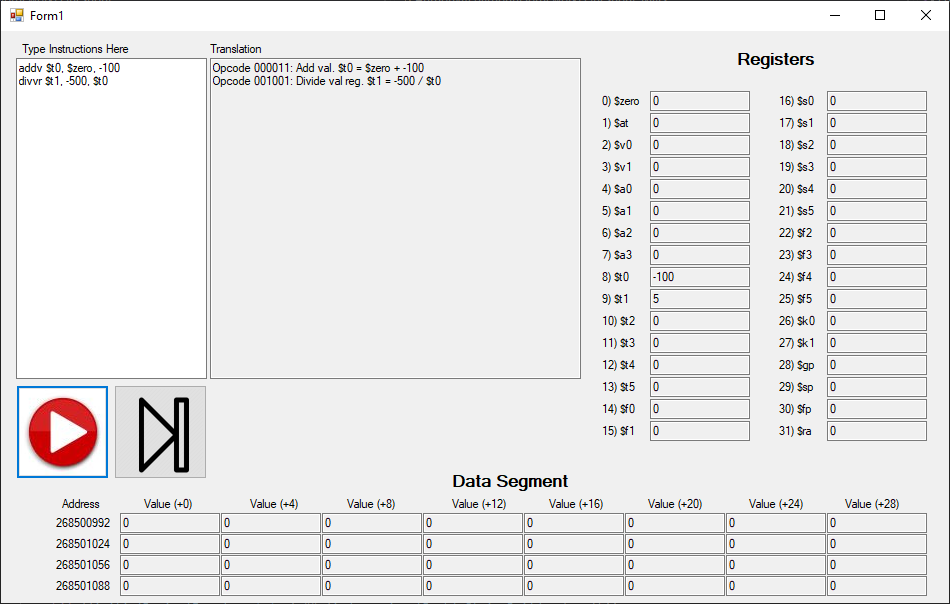


Divvr: Opcode = 001001

Divide a 16-bit value by a register. Compatible with floating point and non-floating-point registers.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Bits** | 26-31 | 21-25 | 16-20 | 0-15 |
| **Meaning** | Opcode | Rs | Rt | Signed value |

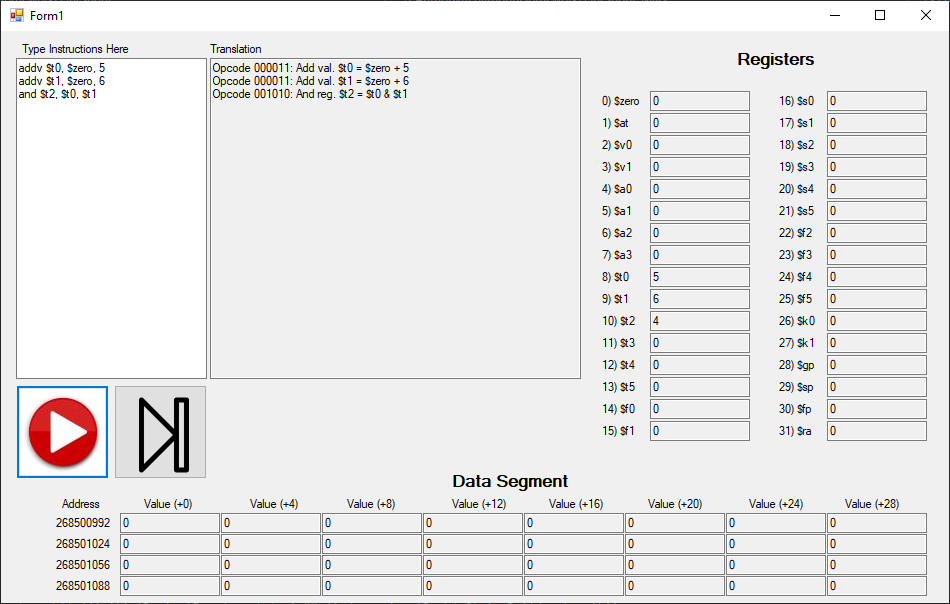


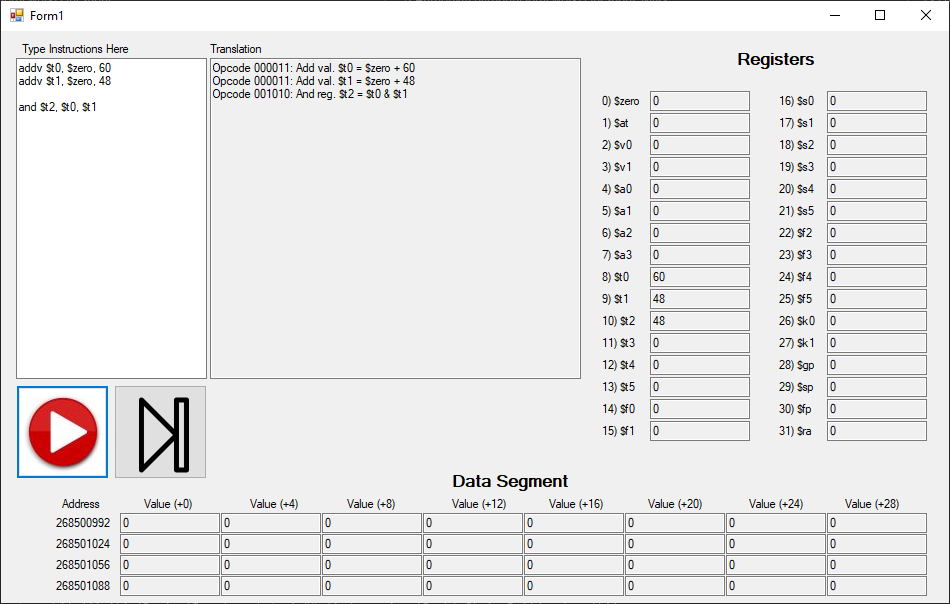


And: Opcode = 001010

Bitwise AND operation.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Bits** | 26-31 | 21-25 | 16-20 | 11-15 | 0-10 |
| **Meaning** | Opcode | Rs | Rt | Rd | 00000000000 |

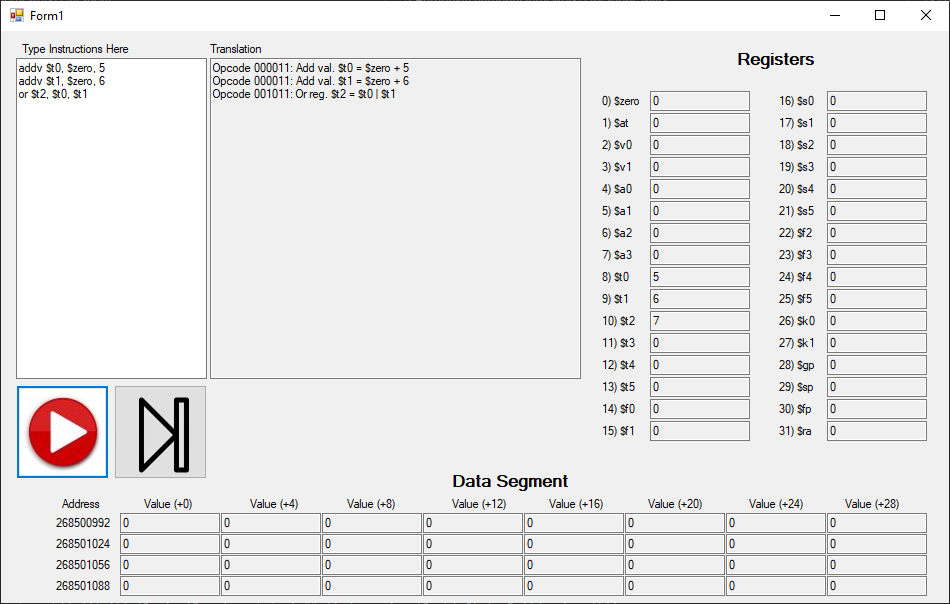


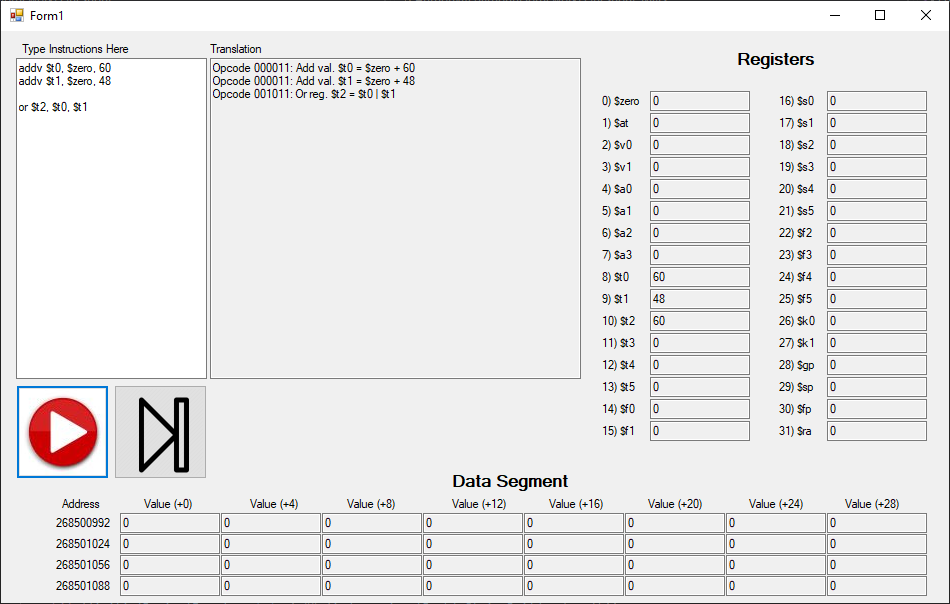


Or: Opcode = 001011

Bitwise OR operation.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Bits** | 26-31 | 21-25 | 16-20 | 11-15 | 0-10 |
| **Meaning** | Opcode | Rs | Rt | Rd | 00000000000 |

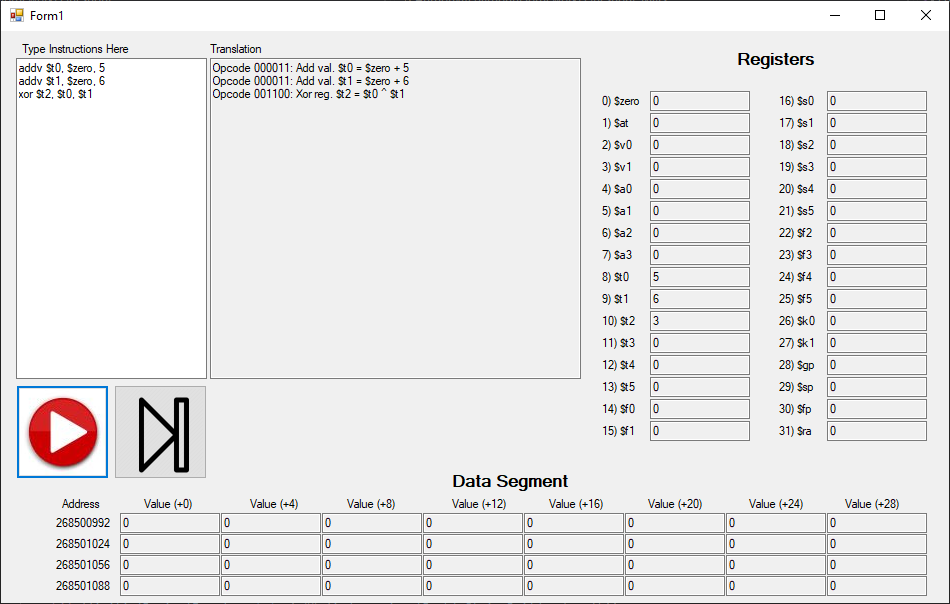


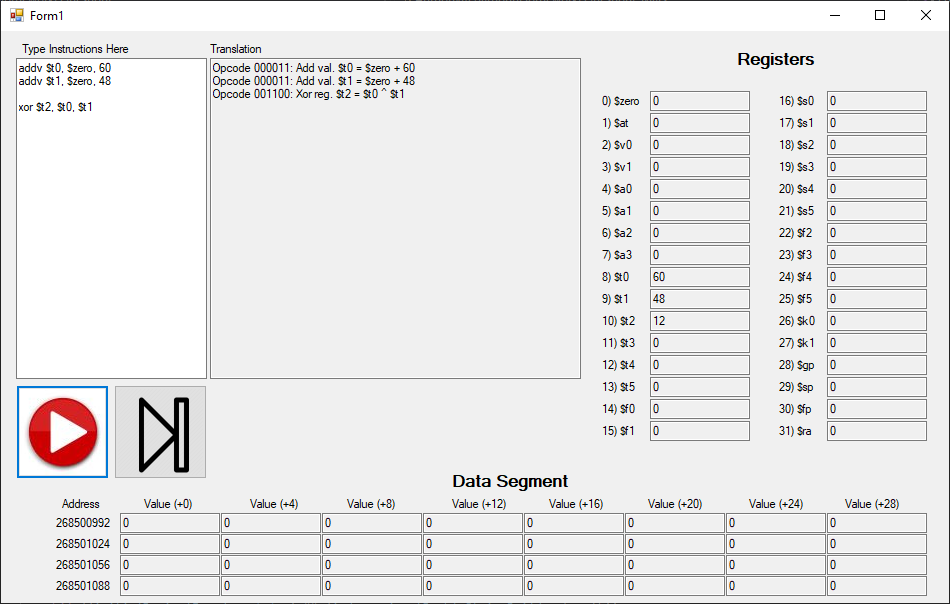


Xor: Opcode = 001100

Bitwise XOR operation.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Bits** | 26-31 | 21-25 | 16-20 | 11-15 | 0-10 |
| **Meaning** | Opcode | Rs | Rt | Rd | 00000000000 |

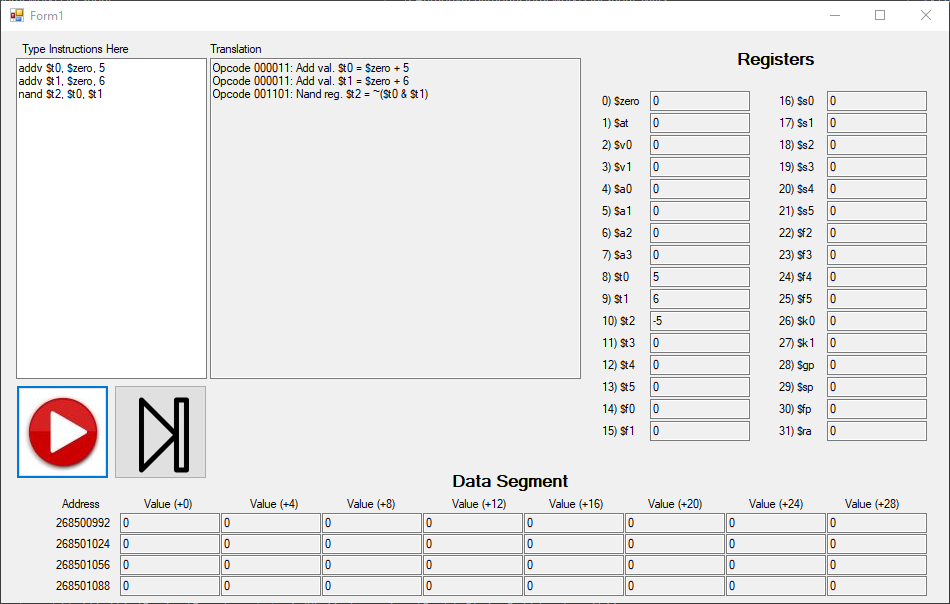


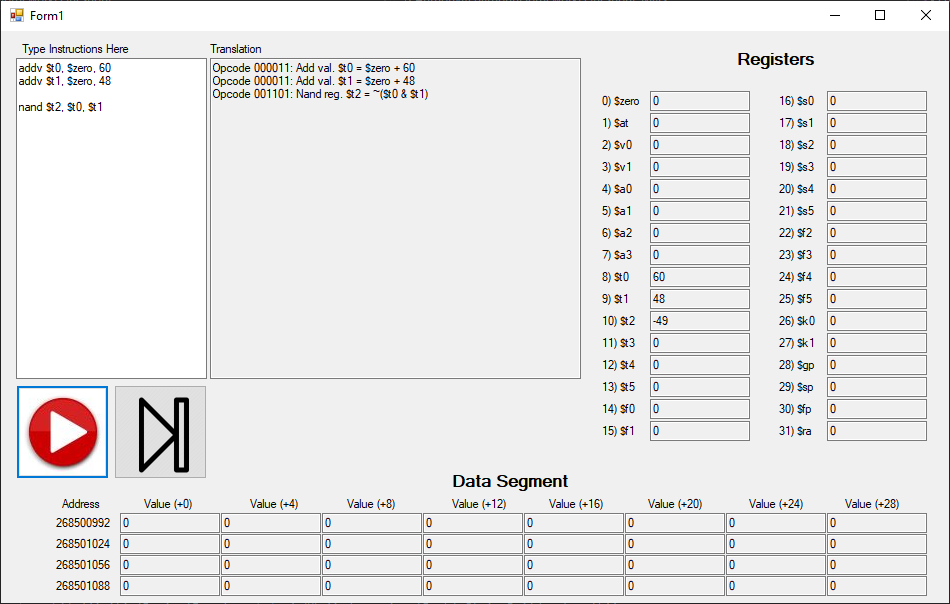


Nand: Opcode = 001101

Bitwise NAND operation.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Bits** | 26-31 | 21-25 | 16-20 | 11-15 | 0-10 |
| **Meaning** | Opcode | Rs | Rt | Rd | 00000000000 |

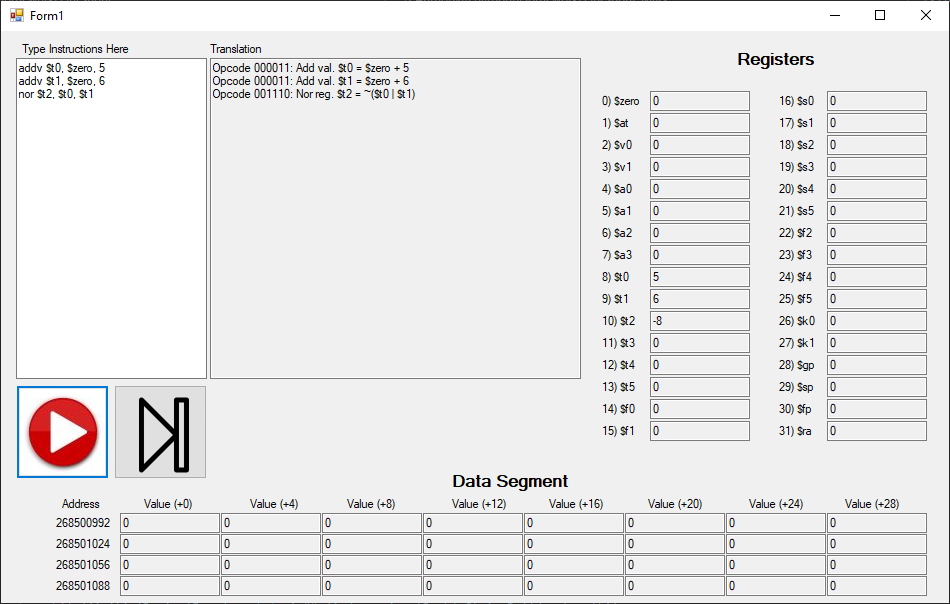


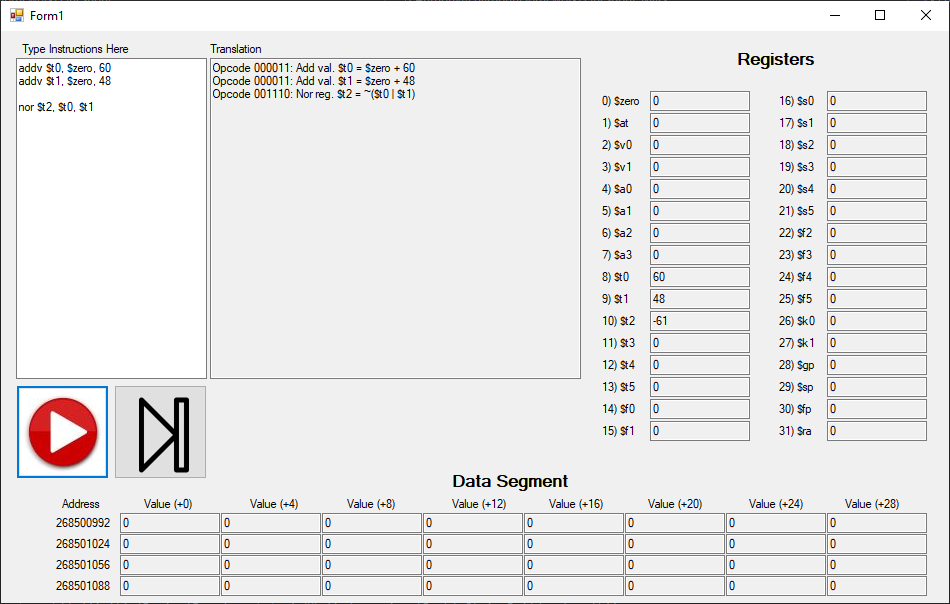


Nor: Opcode = 001110

Bitwise NOR operation.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Bits** | 26-31 | 21-25 | 16-20 | 11-15 | 0-10 |
| **Meaning** | Opcode | Rs | Rt | Rd | 00000000000 |

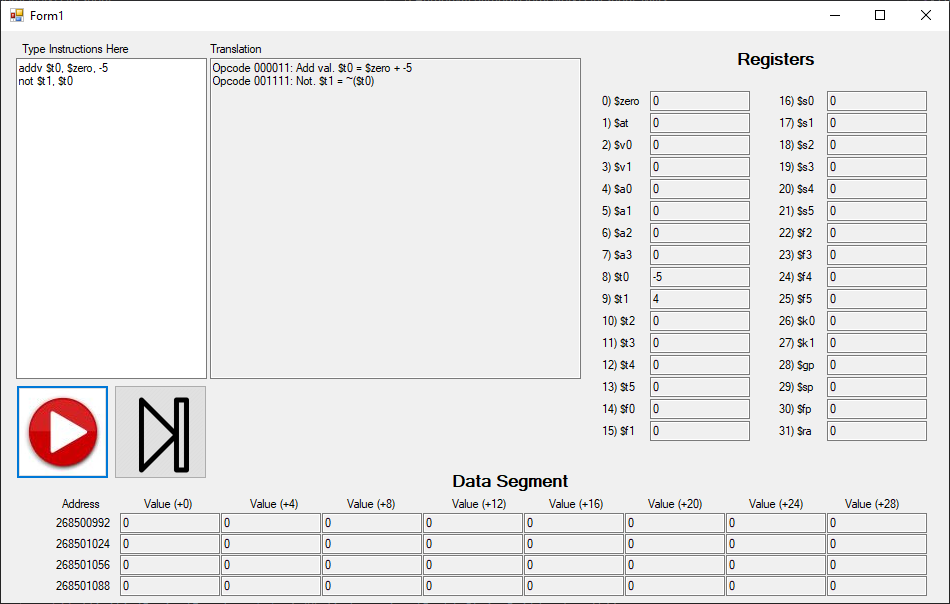


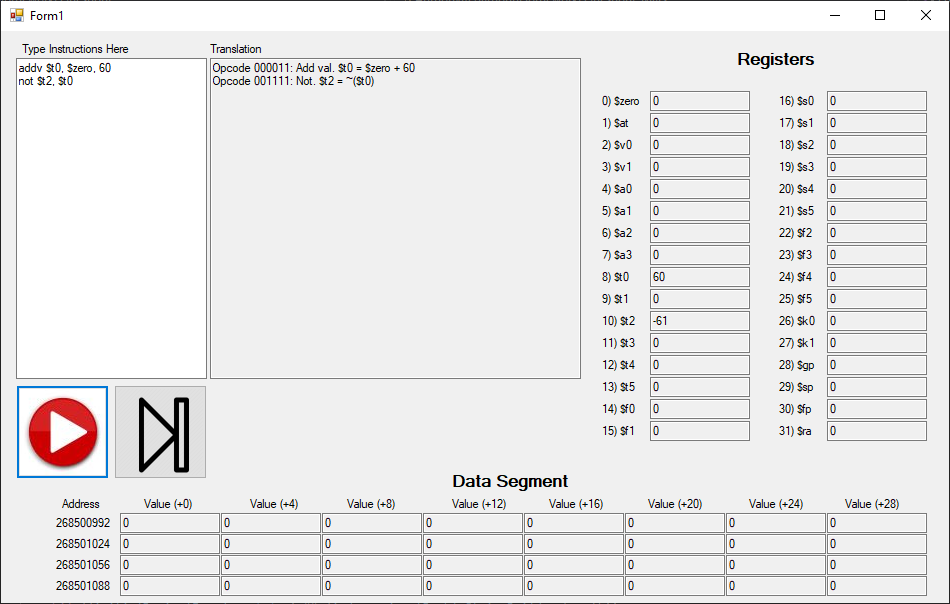


Not: Opcode = 001111

Bitwise NOT operation.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Bits** | 26-31 | 21-25 | 16-20 | 0-15 |
| **Meaning** | Opcode | Rs | Rt | 0x0000 |

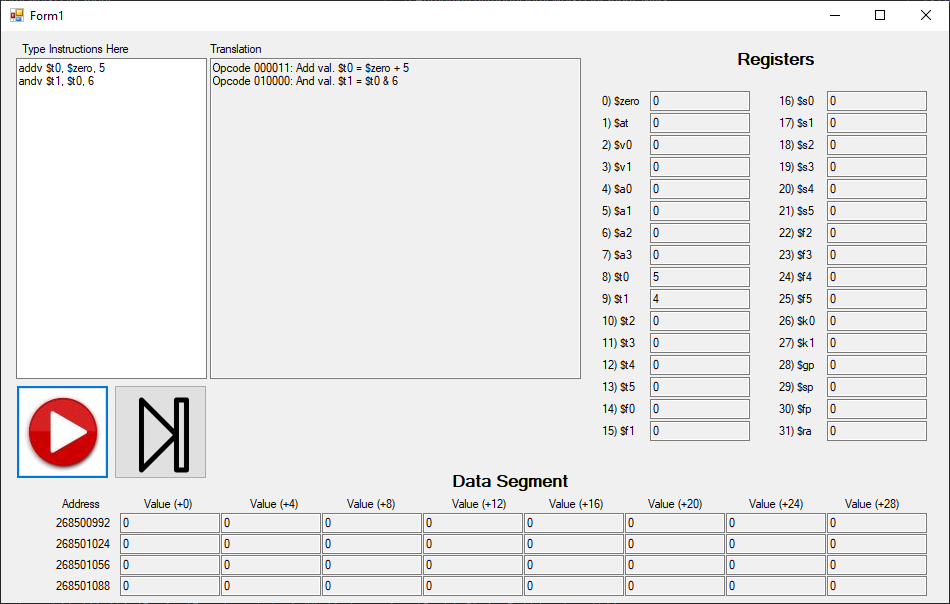


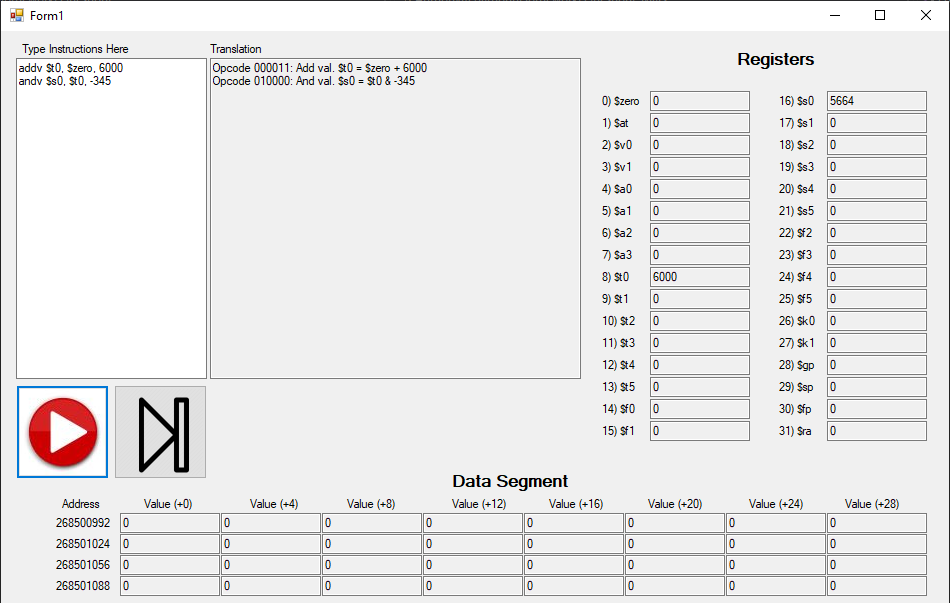


Andv: Opcode = 010000

Bitwise AND operation with a register and a 16-bit value.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Bits** | 26-31 | 21-25 | 16-20 | 0-15 |
| **Meaning** | Opcode | Rs | Rt | Signed value |

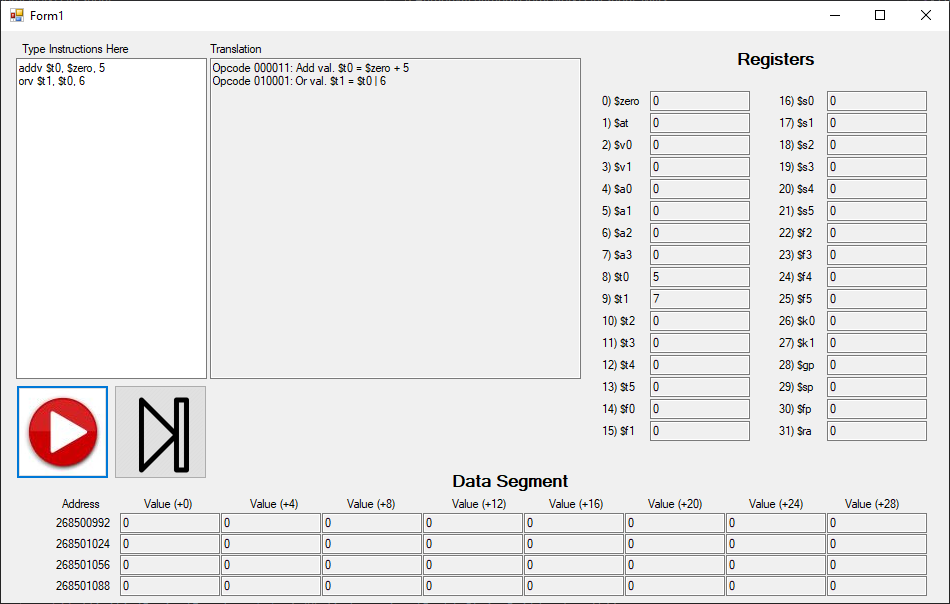


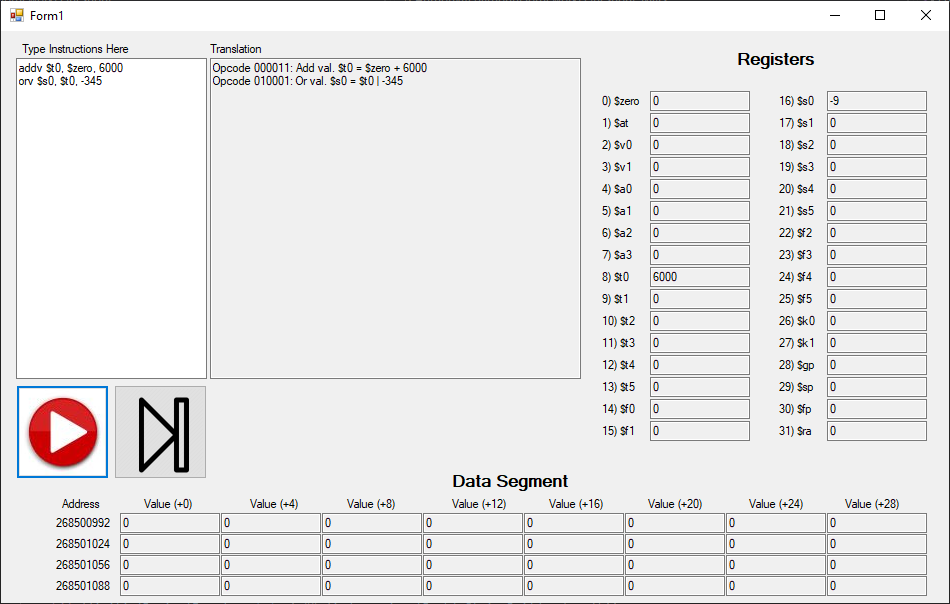


Orv: Opcode = 010001

Bitwise OR operation with a register and a 16-bit value.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Bits** | 26-31 | 21-25 | 16-20 | 0-15 |
| **Meaning** | Opcode | Rs | Rt | Signed value |

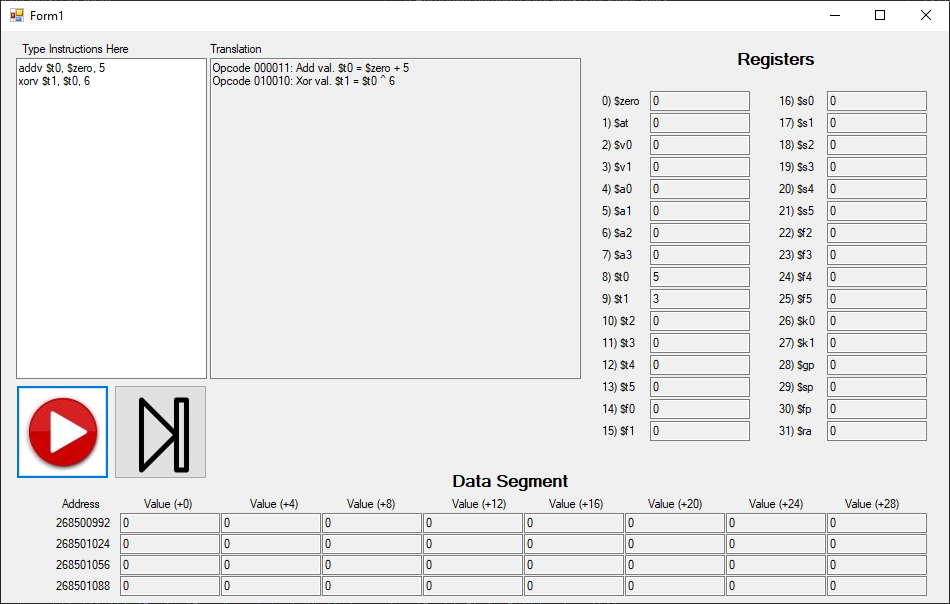


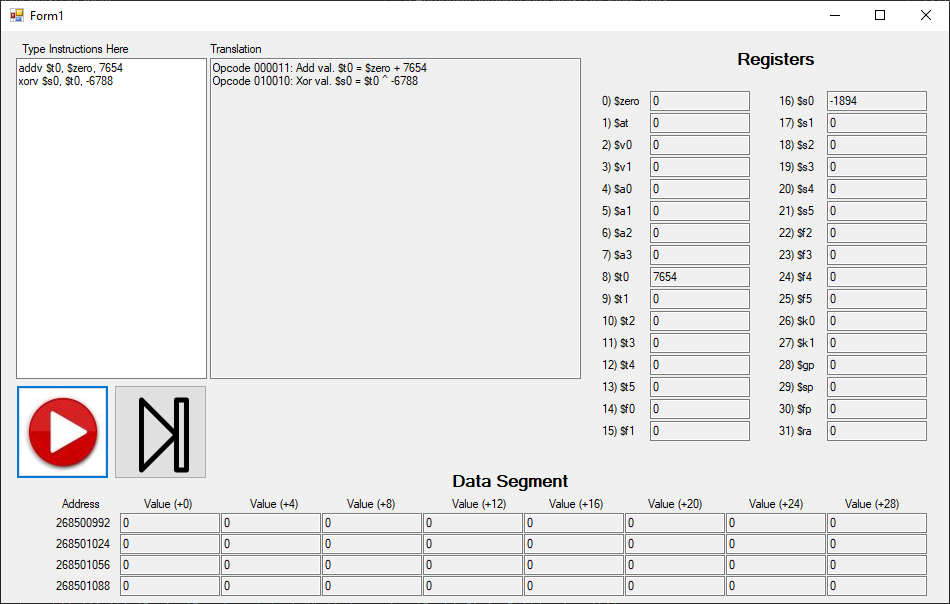


Xorv: Opcode = 010010

Bitwise XOR operation with a register and a 16-bit value.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Bits** | 26-31 | 21-25 | 16-20 | 0-15 |
| **Meaning** | Opcode | Rs | Rt | Signed value |

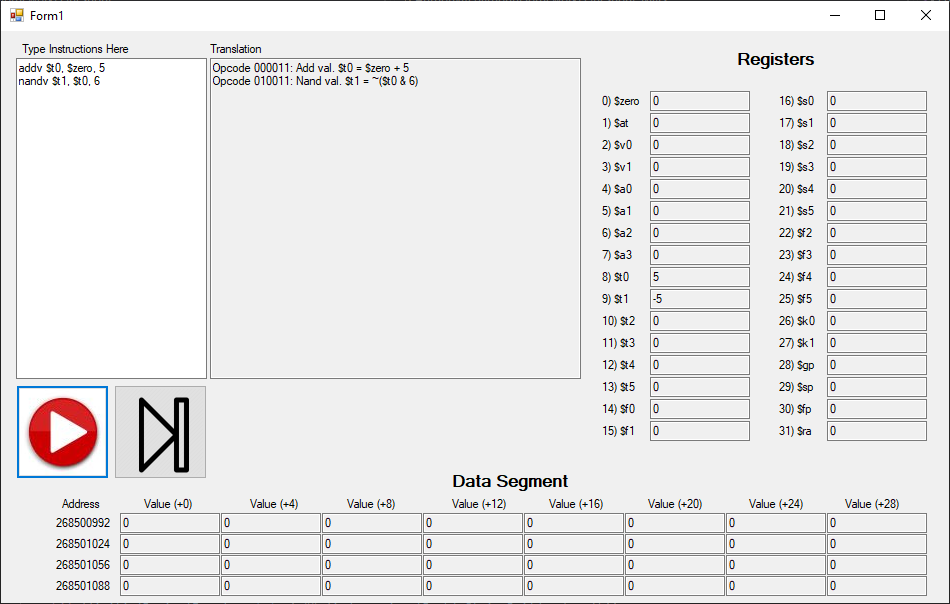


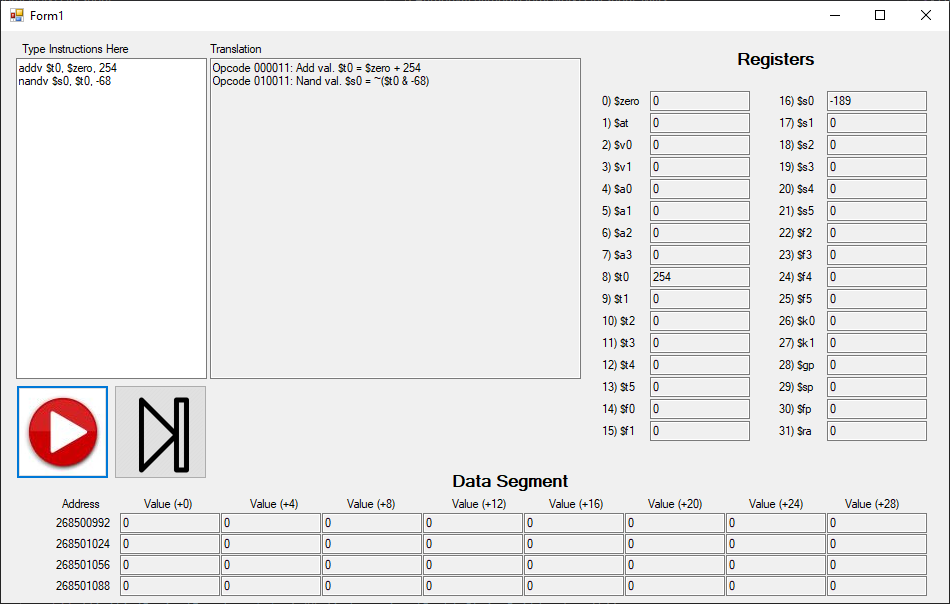


Nandv: Opcode = 010011

Bitwise NAND operation with a register and a 16-bit value.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Bits** | 26-31 | 21-25 | 16-20 | 0-15 |
| **Meaning** | Opcode | Rs | Rt | Signed value |

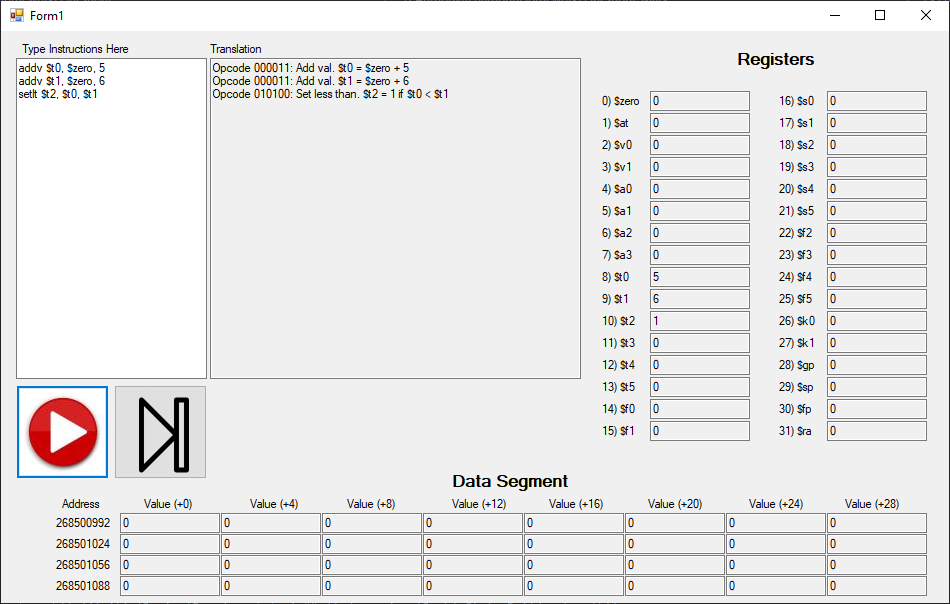


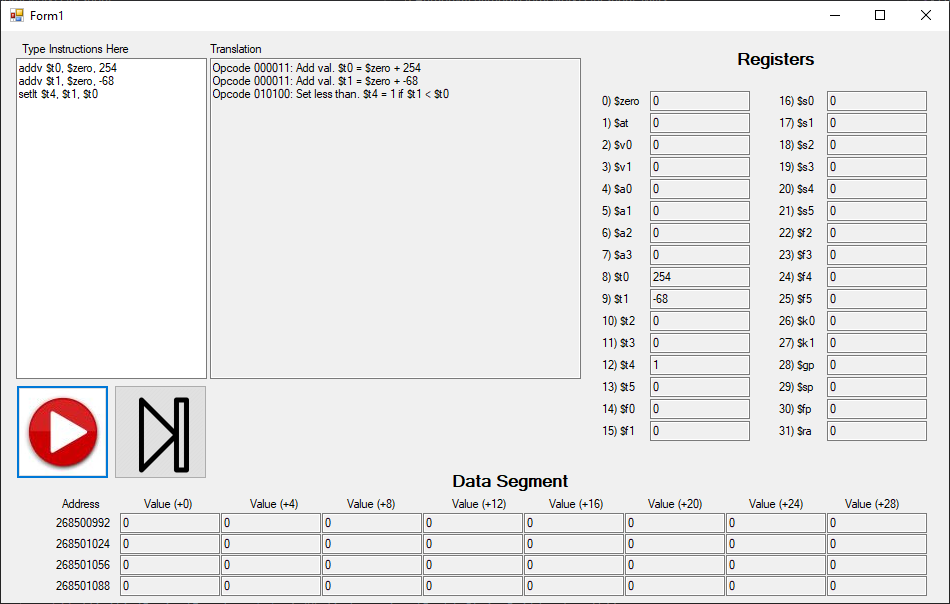


Setlt: Opcode = 010100

Set Rt to 1 if Rs is less than Rd. Set Rt to 0 otherwise.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Bits** | 26-31 | 21-25 | 16-20 | 11-15 | 0-10 |
| **Meaning** | Opcode | Rs | Rt | Rd | 00000000000 |

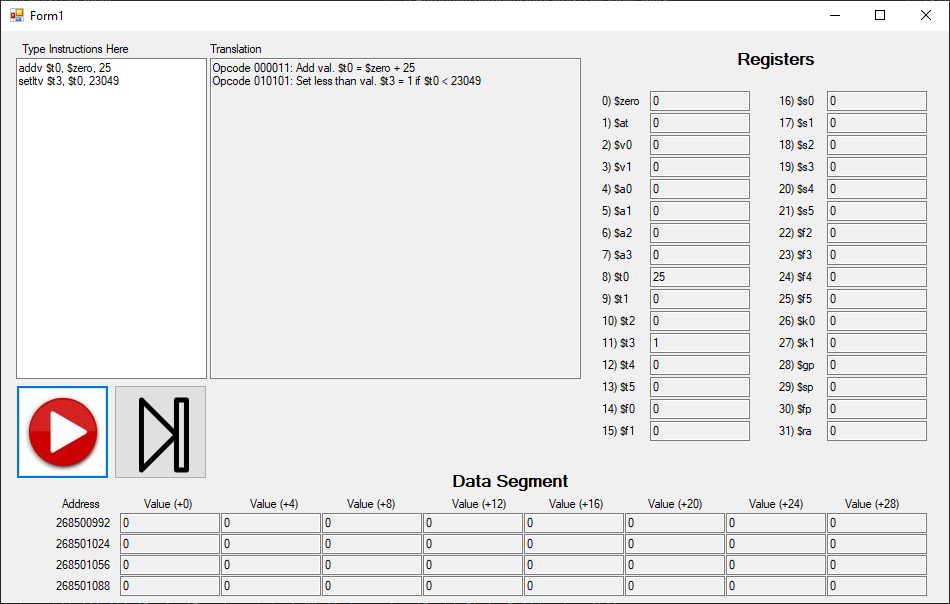




Setltv: Opcode = 010101

Set Rt to 1 if Rs is less than the 16-bit value. Set Rt to 0 otherwise.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Bits** | 26-31 | 21-25 | 16-20 | 0-15 |
| **Meaning** | Opcode | Rs | Rt | Signed value |



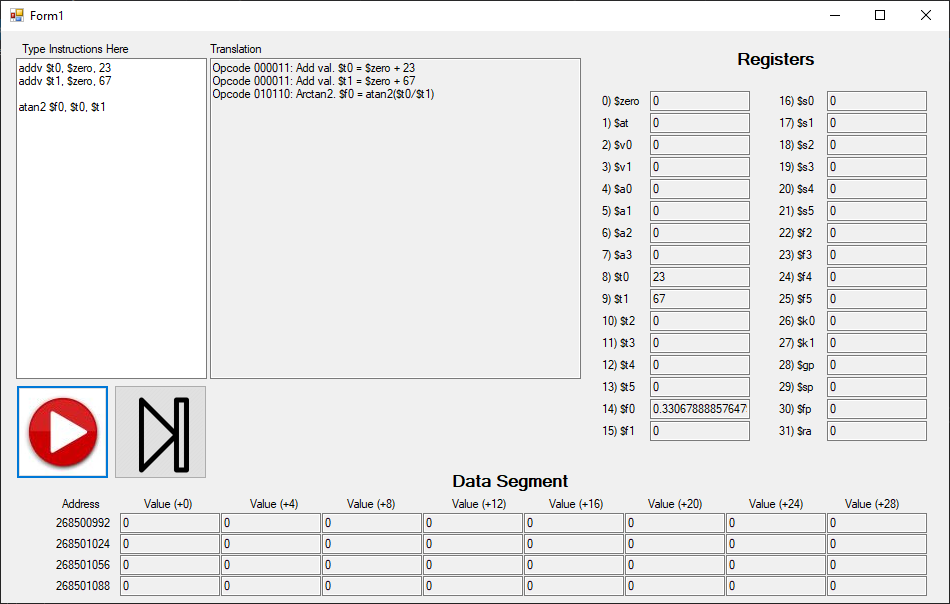


Atan2: Opcode = 010110

Rt equals the atan2 of Rs over Rd.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Bits** | 26-31 | 21-25 | 16-20 | 11-15 | 0-10 |
| **Meaning** | Opcode | Rs | Rt | Rd | 00000000000 |

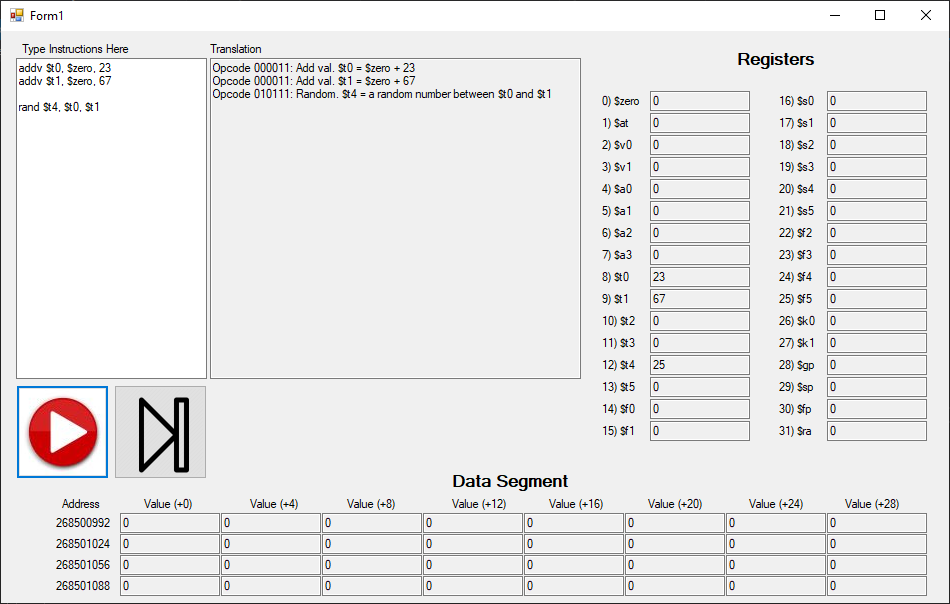


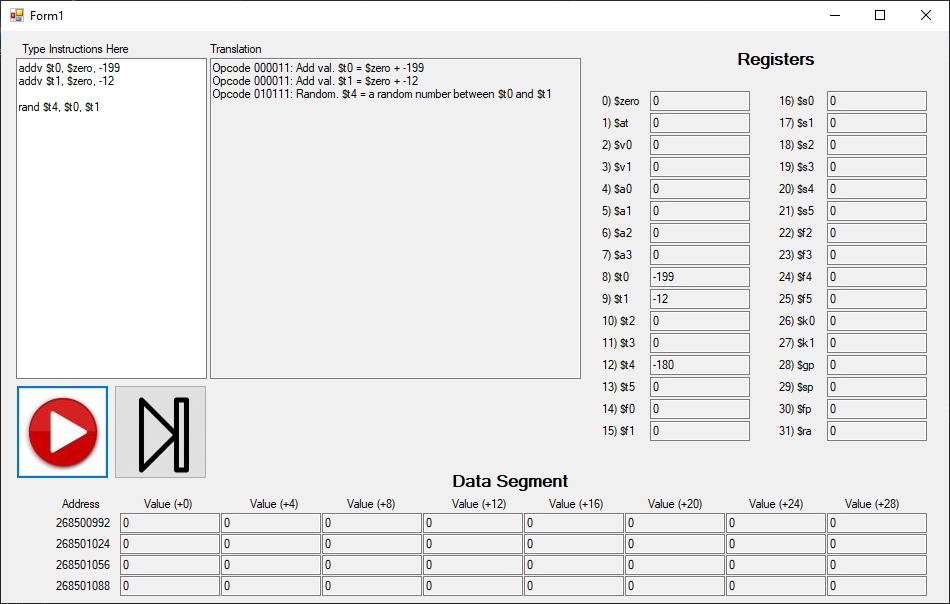


Rand: Opcode = 010111

Rt will be a random number between Rs and Rd (inclusive).

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Bits** | 26-31 | 21-25 | 16-20 | 11-15 | 0-10 |
| **Meaning** | Opcode | Rs | Rt | Rd | 00000000000 |

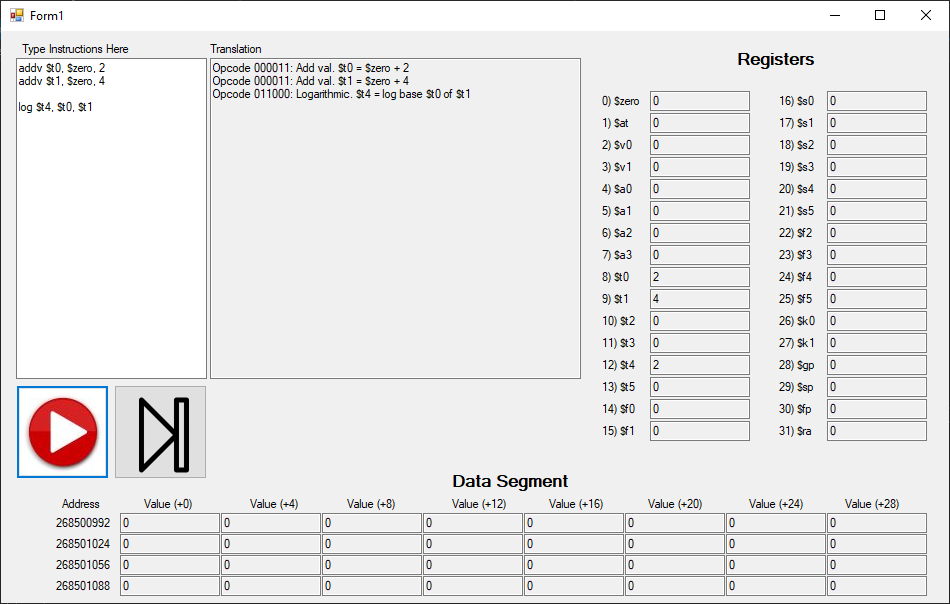


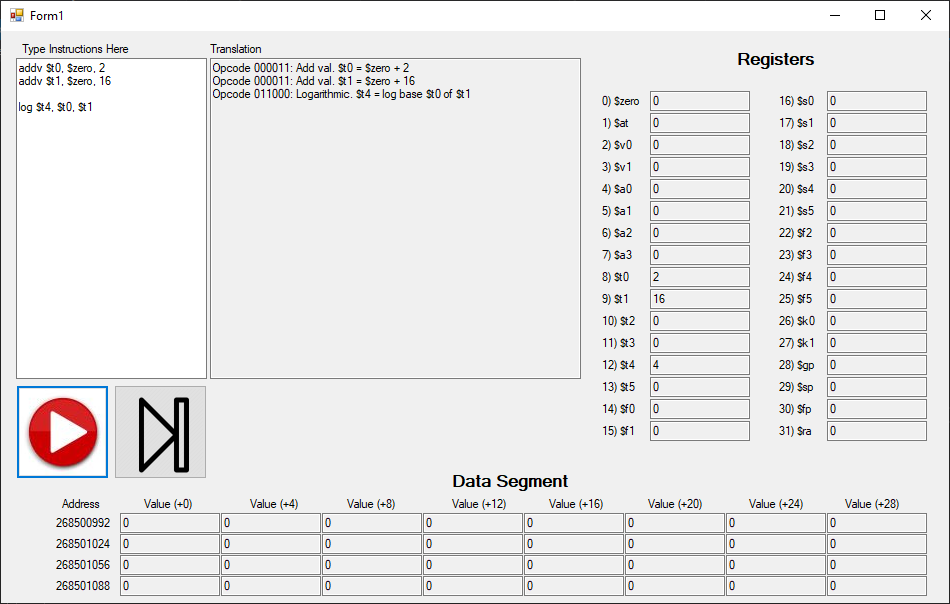


Log: Opcode = 011000

Rt is equal to the logarithm base Rs of Rd.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Bits** | 26-31 | 21-25 | 16-20 | 11-15 | 0-10 |
| **Meaning** | Opcode | Rs | Rt | Rd | 00000000000 |

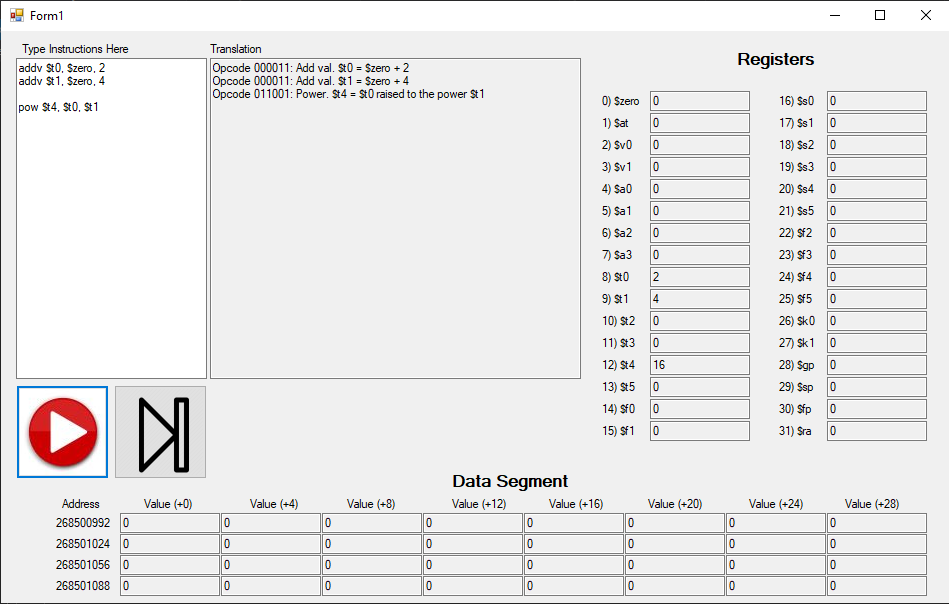


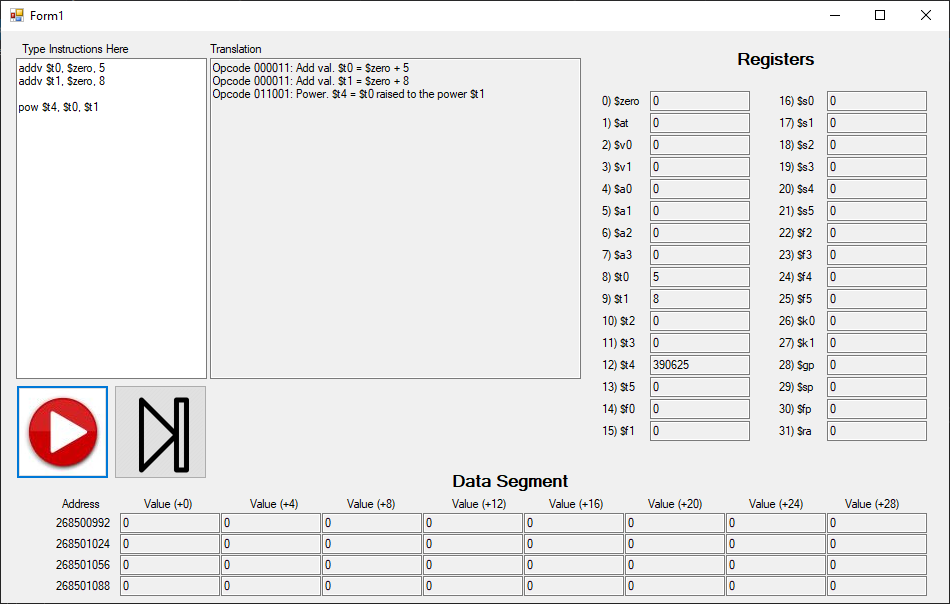


Pow: Opcode = 011001

Rt is equal to Rs raised to the power of Rd.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Bits** | 26-31 | 21-25 | 16-20 | 11-15 | 0-10 |
| **Meaning** | Opcode | Rs | Rt | Rd | 00000000000 |

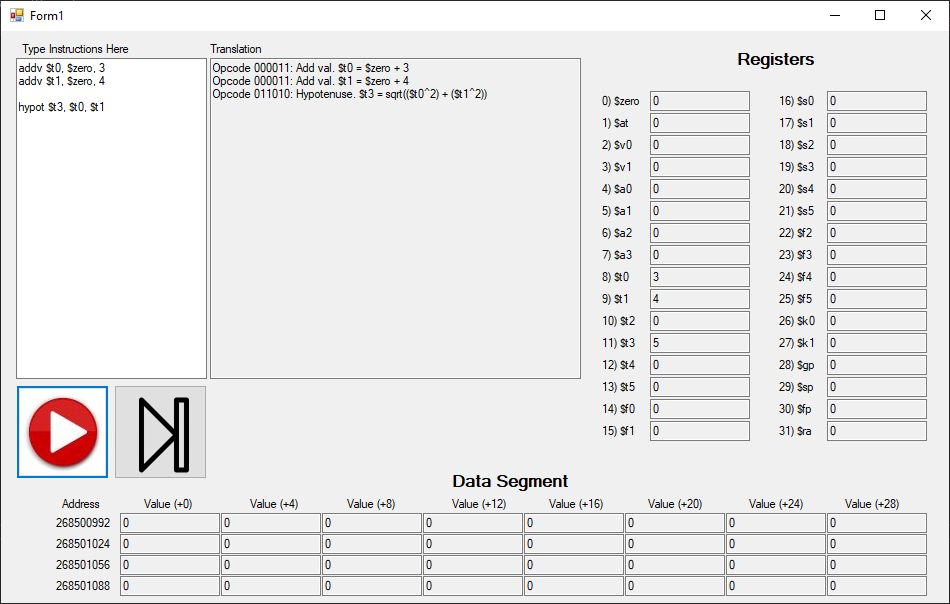


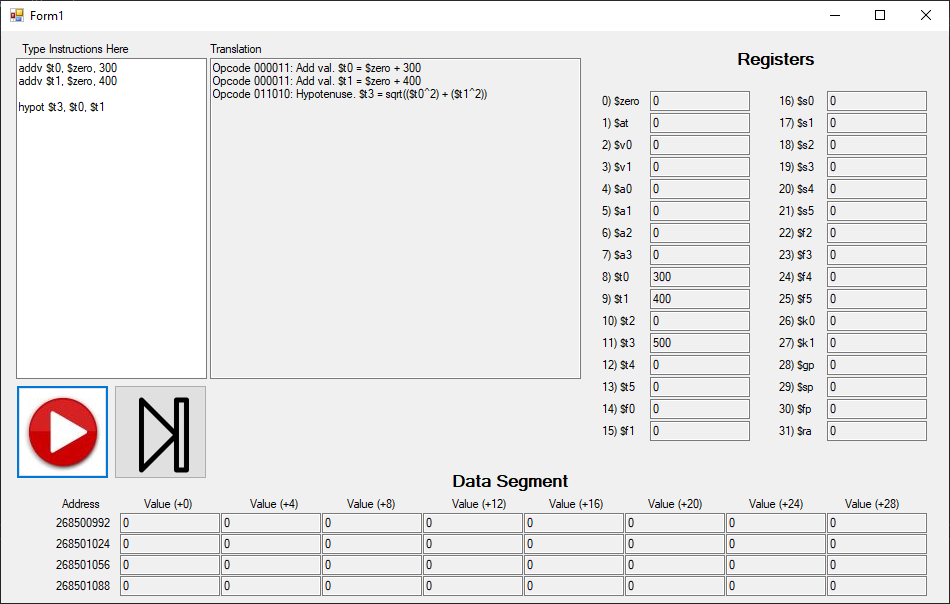


Hypot: Opcode = 011010

Rt is the hypotenuse of a right triangle where the adjacent sides are of length Rs and Rd.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Bits** | 26-31 | 21-25 | 16-20 | 11-15 | 0-10 |
| **Meaning** | Opcode | Rs | Rt | Rd | 00000000000 |

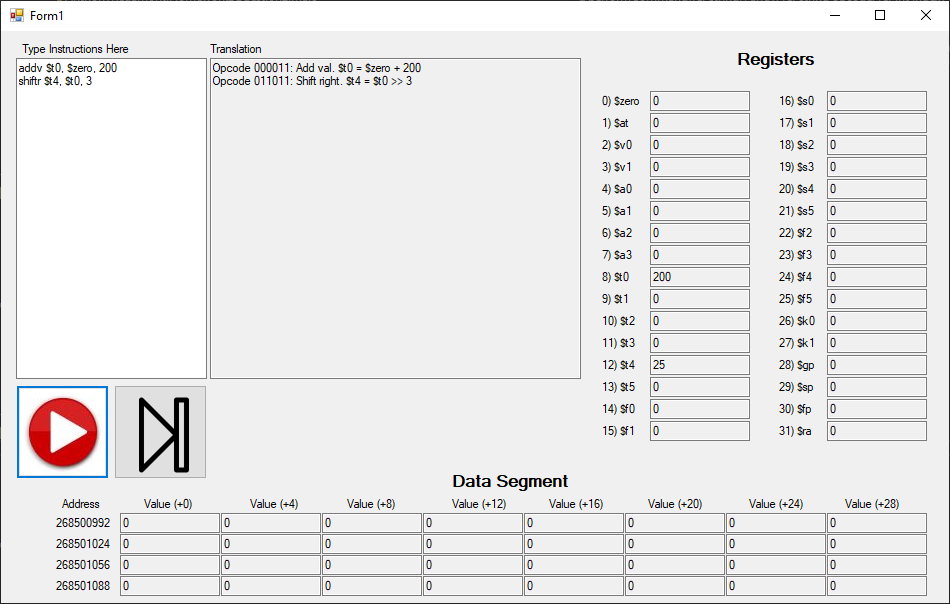


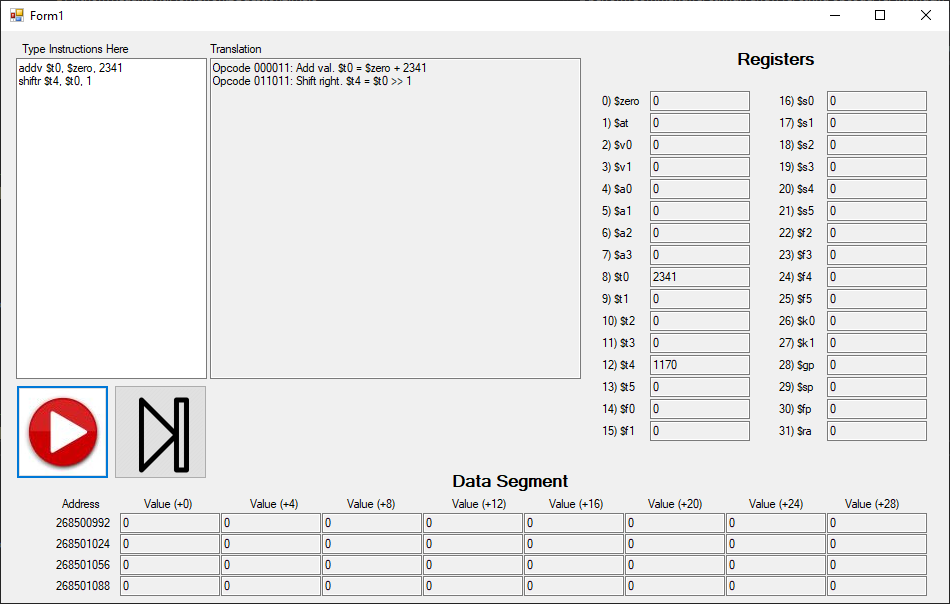


Shiftr: Opcode = 011011

Rt is equal to Rs >> Rd.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Bits** | 26-31 | 21-25 | 16-20 | 0-15 |
| **Meaning** | Opcode | Rs | Rt | Signed value |

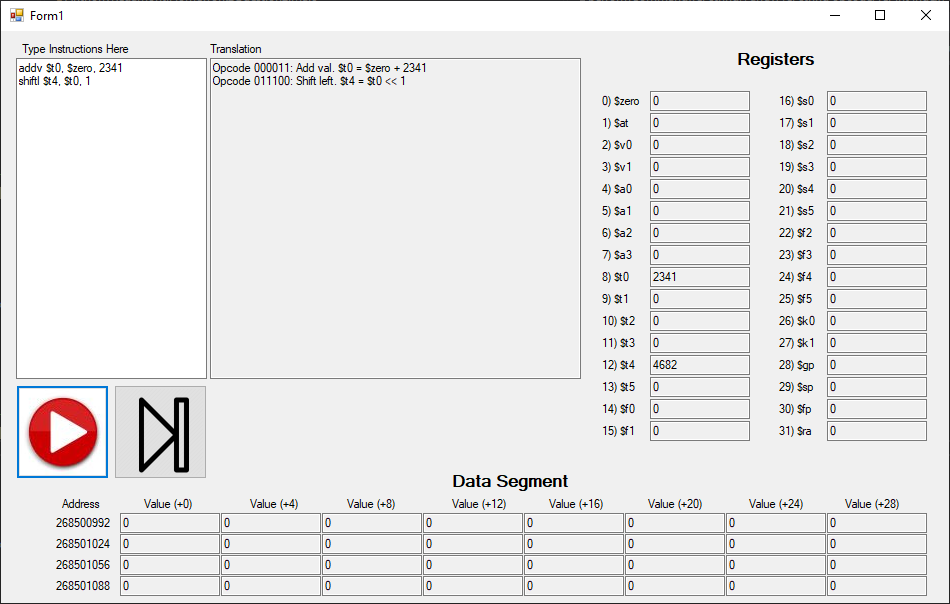


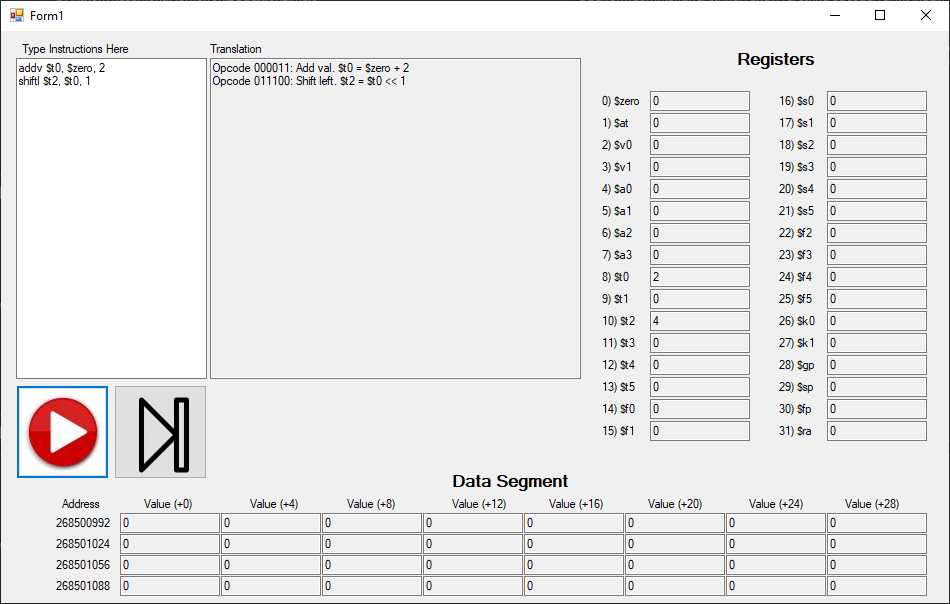


Shiftl: Opcode = 011100

Rt is equal to Rs << Rd.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Bits** | 26-31 | 21-25 | 16-20 | 0-15 |
| **Meaning** | Opcode | Rs | Rt | Signed value |

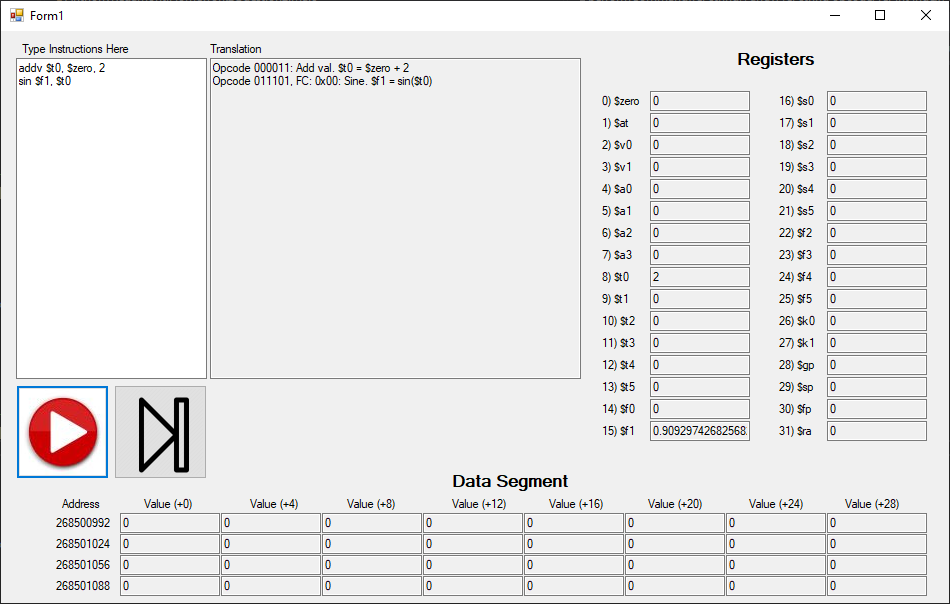


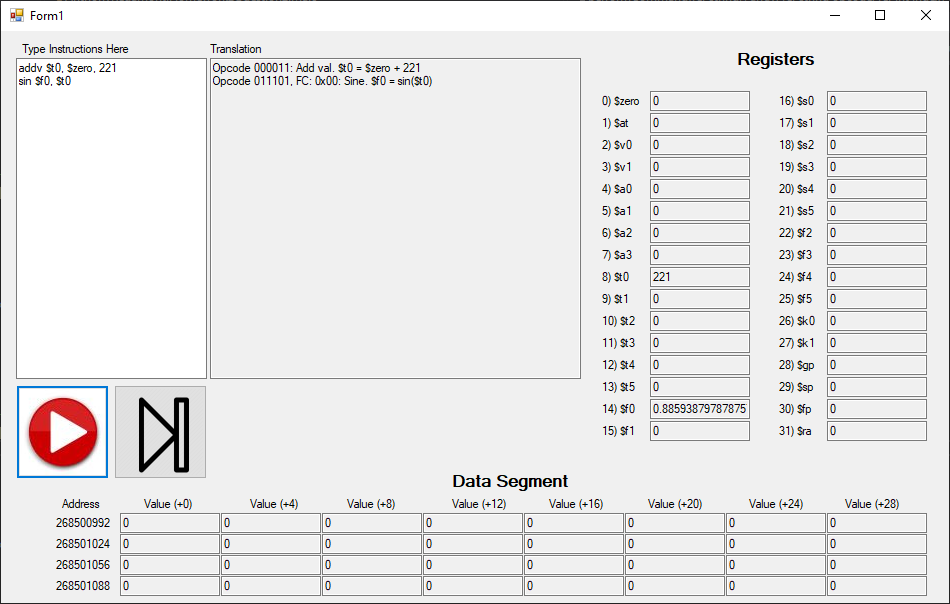


Sin: Opcode = 011101, Function Code = 0x00

Rt is equal to the sine of Rs.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Bits** | 26-31 | 21-25 | 16-20 | 8-15 | 0-7 |
| **Meaning** | Opcode | Rs | Rt | 0x00 | Function Code |

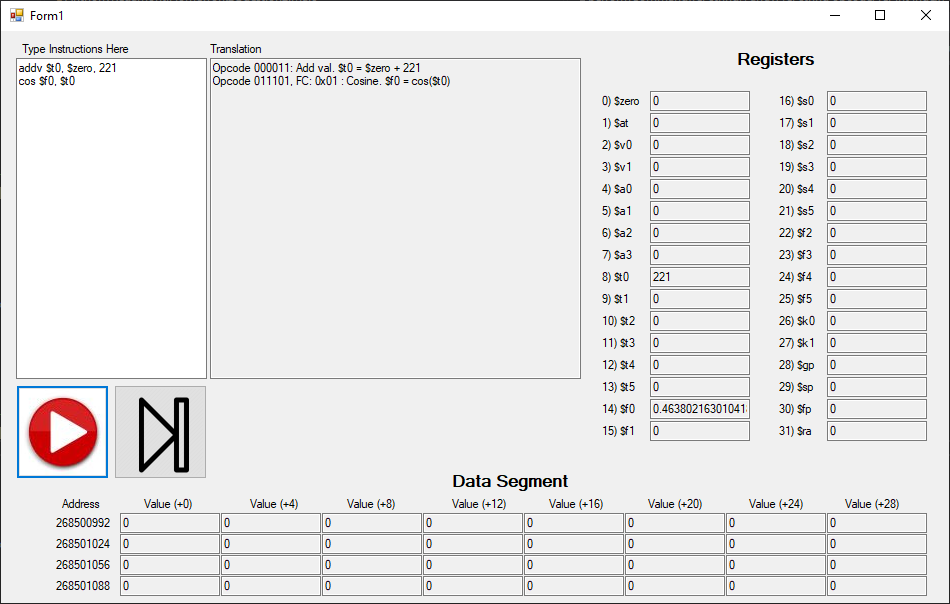


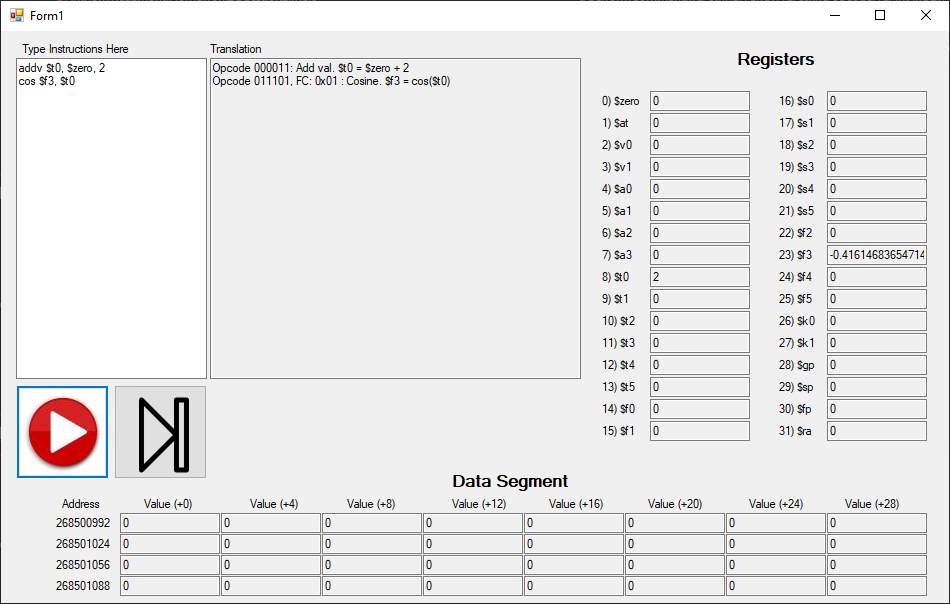


Cos: Opcode = 011101, Function Code = 0x01

Rt is equal to the cosine of Rs.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Bits** | 26-31 | 21-25 | 16-20 | 8-15 | 0-7 |
| **Meaning** | Opcode | Rs | Rt | 0x00 | Function Code |

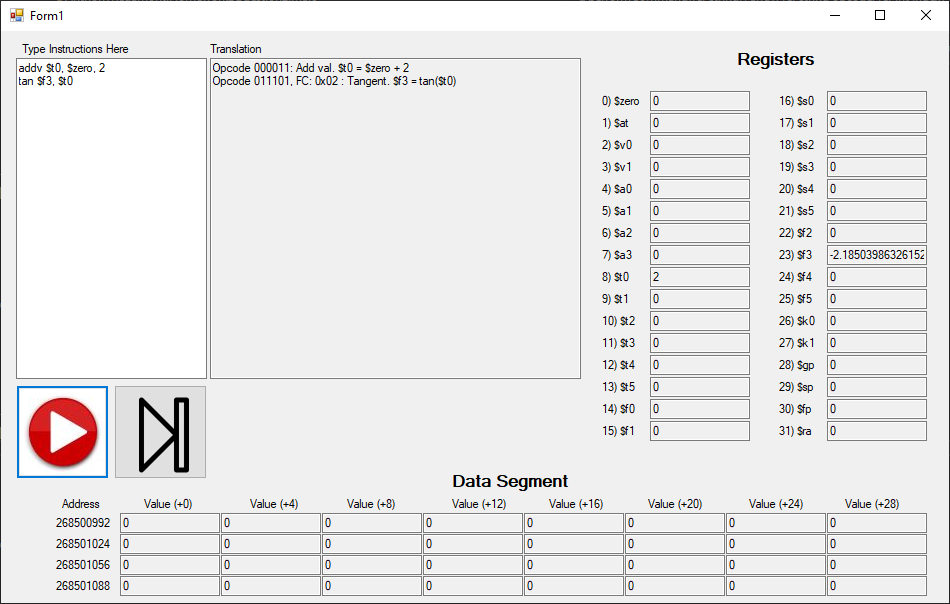


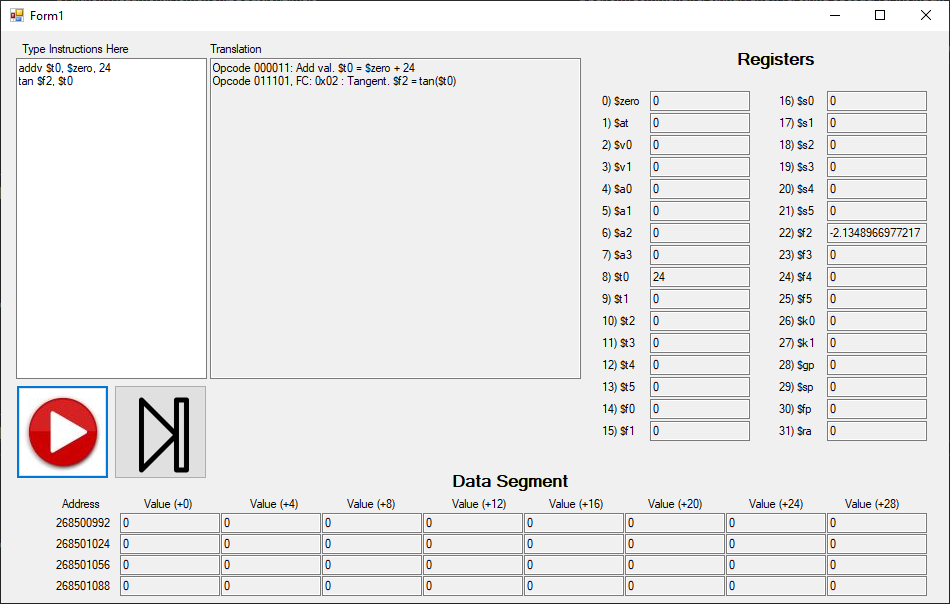


Tan: Opcode = 011101, Function Code = 0x02

Rt is equal to the tangent of Rs.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Bits** | 26-31 | 21-25 | 16-20 | 8-15 | 0-7 |
| **Meaning** | Opcode | Rs | Rt | 0x00 | Function Code |

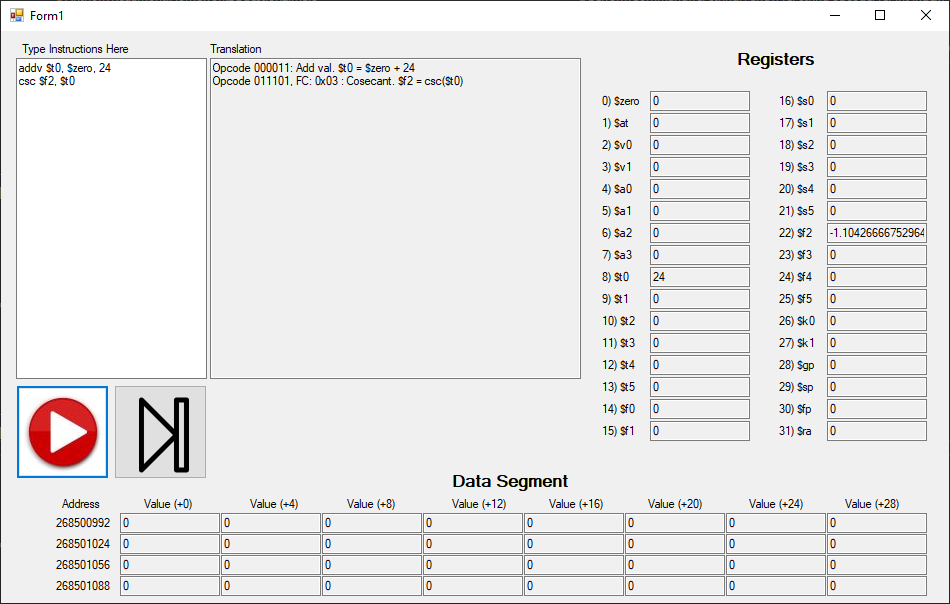


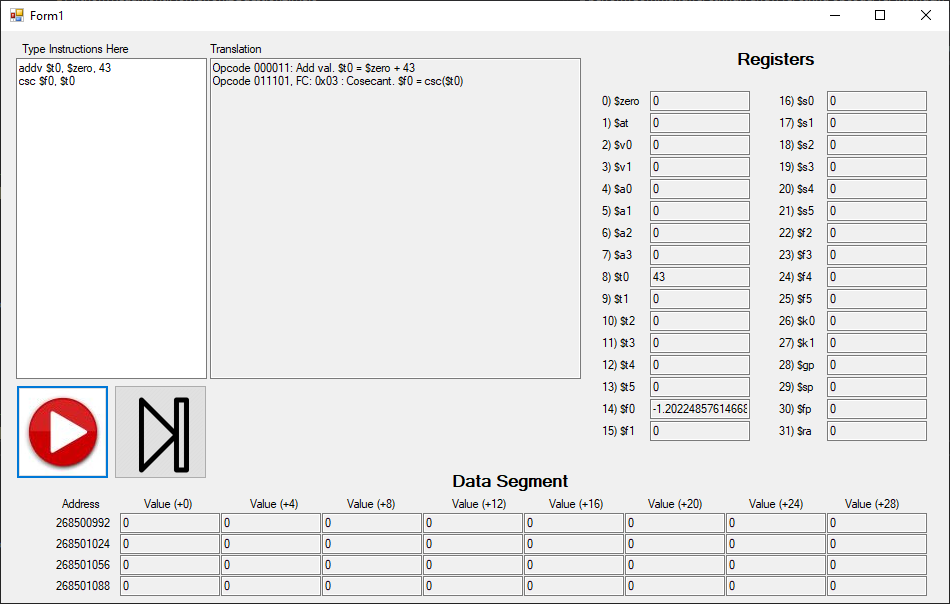


Csc: Opcode = 011101, Function Code = 0x03

Rt is equal to the cosecant of Rs.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Bits** | 26-31 | 21-25 | 16-20 | 8-15 | 0-7 |
| **Meaning** | Opcode | Rs | Rt | 0x00 | Function Code |

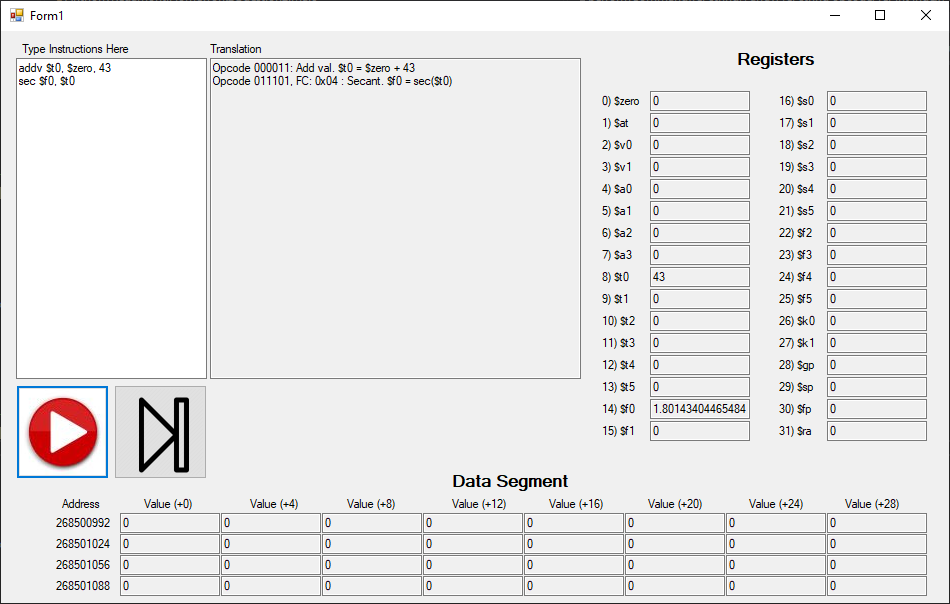


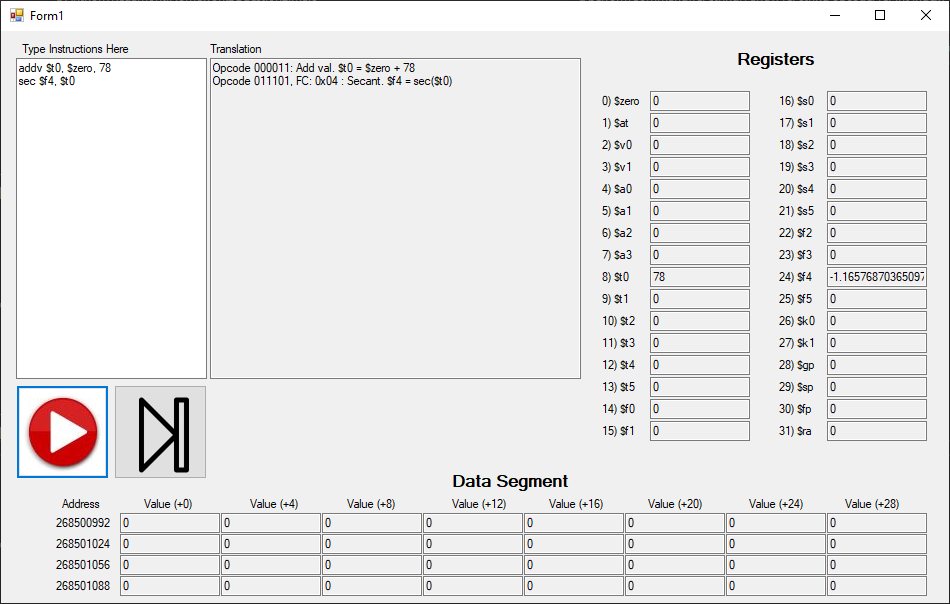


Sec: Opcode = 011101, Function Code = 0x04

Rt is equal to the secant of Rs.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Bits** | 26-31 | 21-25 | 16-20 | 8-15 | 0-7 |
| **Meaning** | Opcode | Rs | Rt | 0x00 | Function Code |

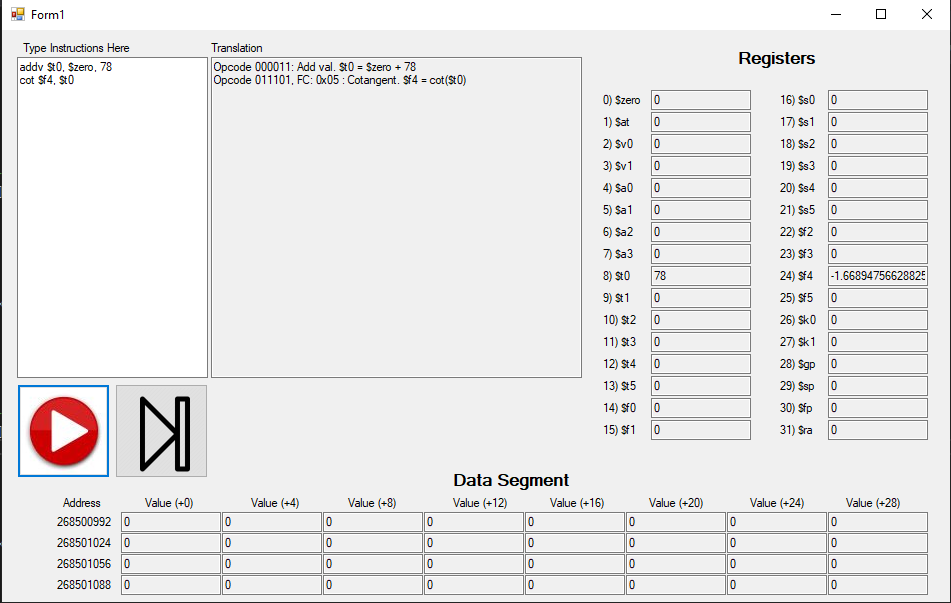


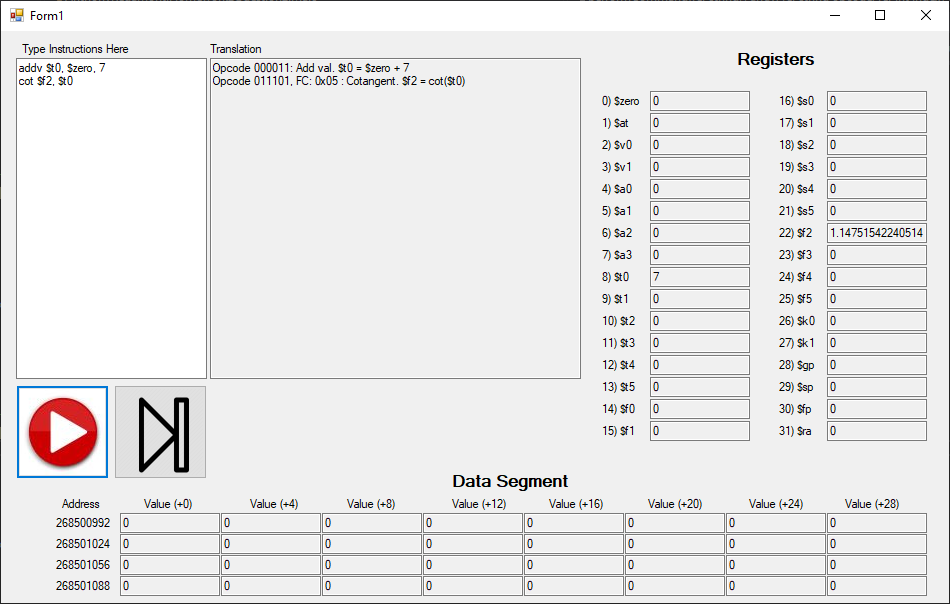


Cot: Opcode = 011101, Function Code = 0x05

Rt is equal to the cotangent of Rs.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Bits** | 26-31 | 21-25 | 16-20 | 8-15 | 0-7 |
| **Meaning** | Opcode | Rs | Rt | 0x00 | Function Code |

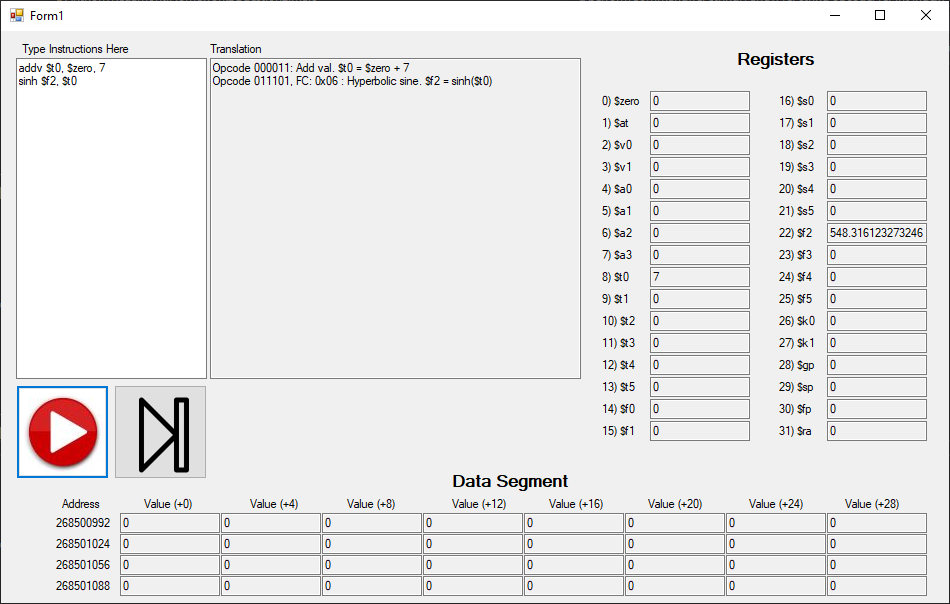


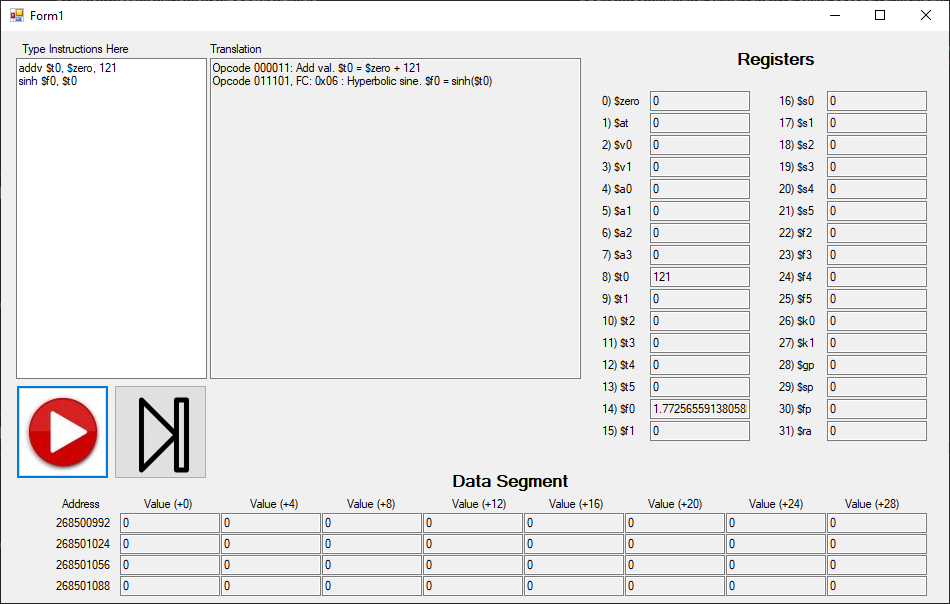


Sinh: Opcode = 011101, Function Code = 0x06

Rt is equal to the hyperbolic sine of Rs.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Bits** | 26-31 | 21-25 | 16-20 | 8-15 | 0-7 |
| **Meaning** | Opcode | Rs | Rt | 0x00 | Function Code |

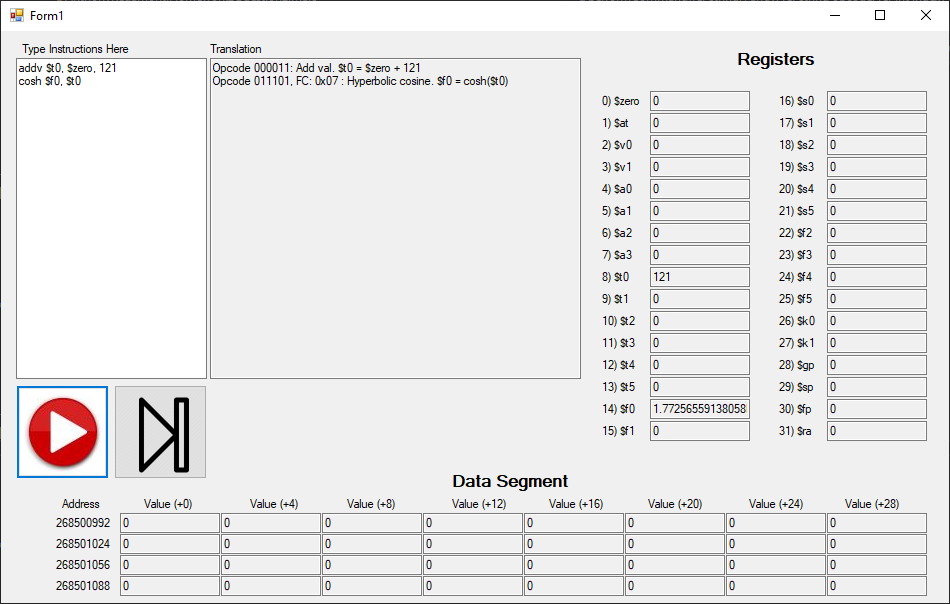


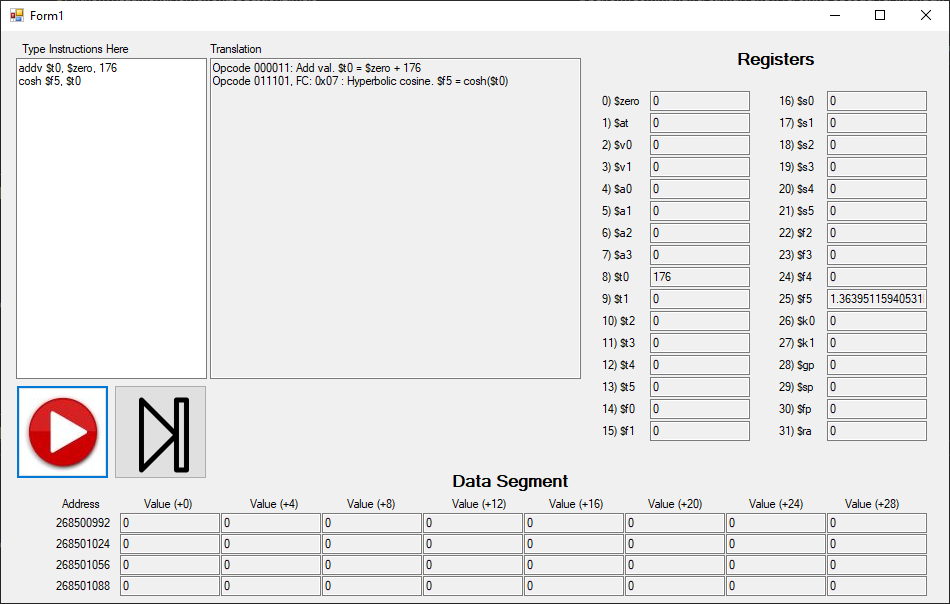


Cosh: Opcode = 011101, Function Code = 0x07

Rt is equal to the hyperbolic cosine of Rs.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Bits** | 26-31 | 21-25 | 16-20 | 8-15 | 0-7 |
| **Meaning** | Opcode | Rs | Rt | 0x00 | Function Code |

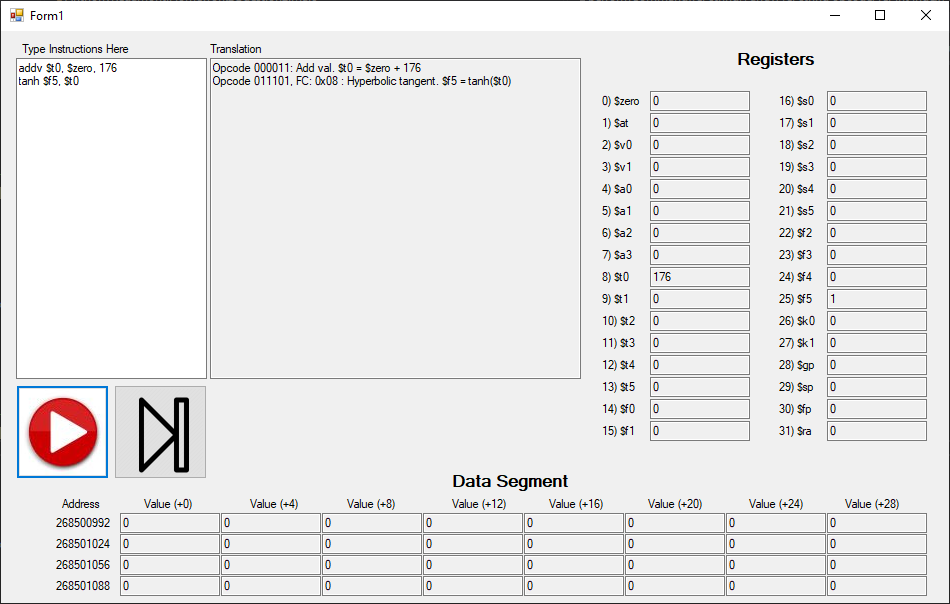


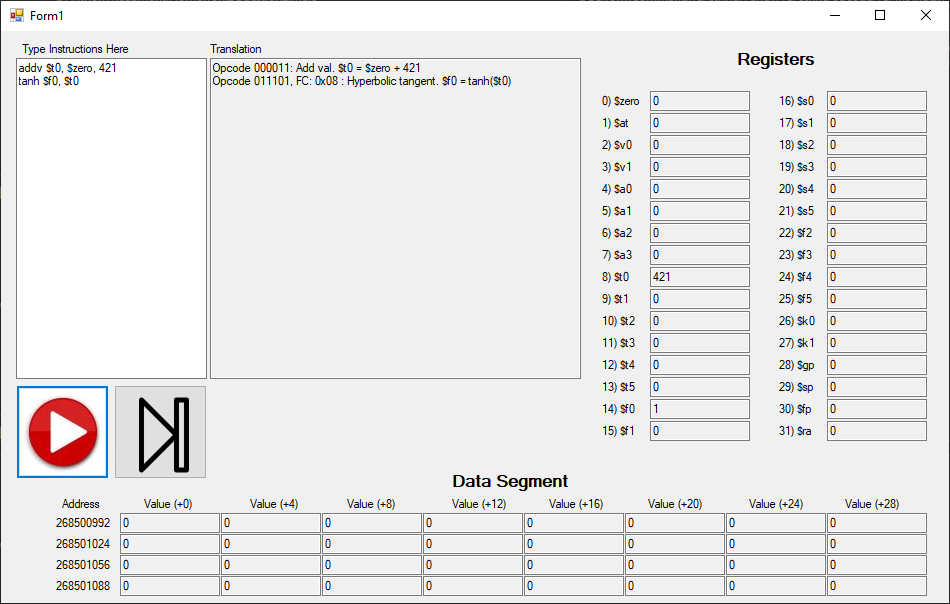


Tanh: Opcode = 011101, Function Code = 0x08

Rt is equal to the hyperbolic tangent of Rs.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Bits** | 26-31 | 21-25 | 16-20 | 8-15 | 0-7 |
| **Meaning** | Opcode | Rs | Rt | 0x00 | Function Code |





Asin: Opcode = 011101, Function Code = 0x09

Rt is equal to the arcsine of Rs.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Bits** | 26-31 | 21-25 | 16-20 | 8-15 | 0-7 |
| **Meaning** | Opcode | Rs | Rt | 0x00 | Function Code |

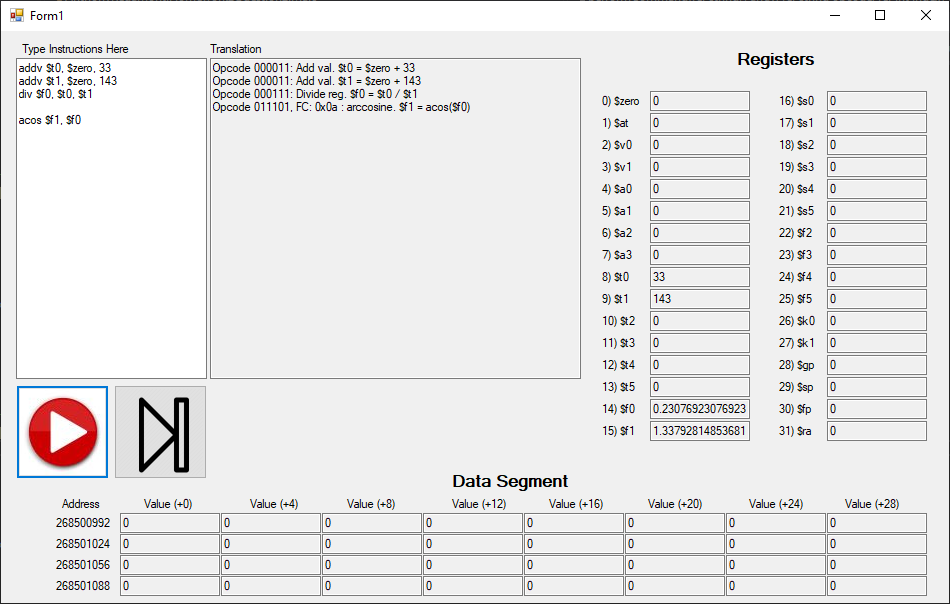


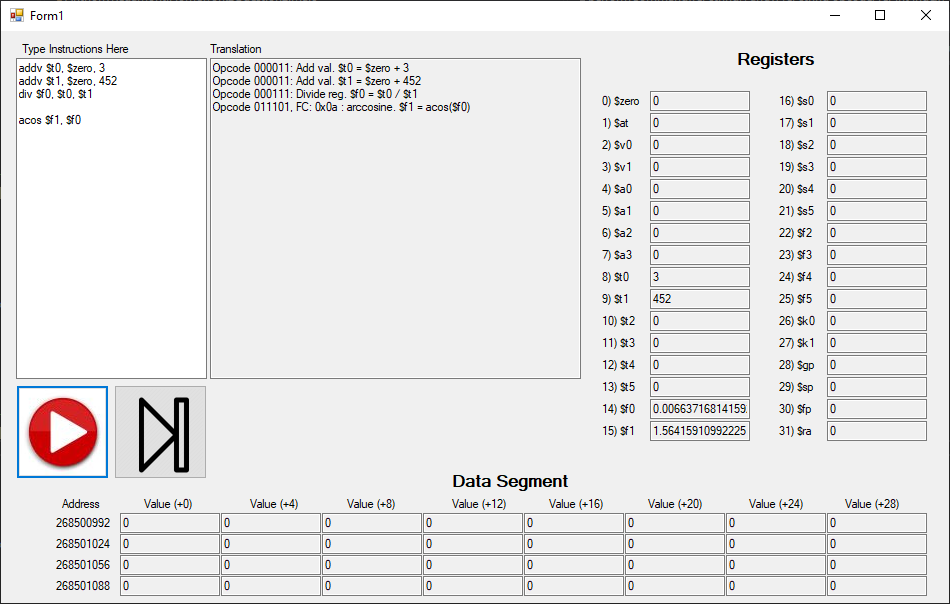


Acos: Opcode = 011101, Function Code = 0x0a

Rt is equal to the arccosine of Rs.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Bits** | 26-31 | 21-25 | 16-20 | 8-15 | 0-7 |
| **Meaning** | Opcode | Rs | Rt | 0x00 | Function Code |

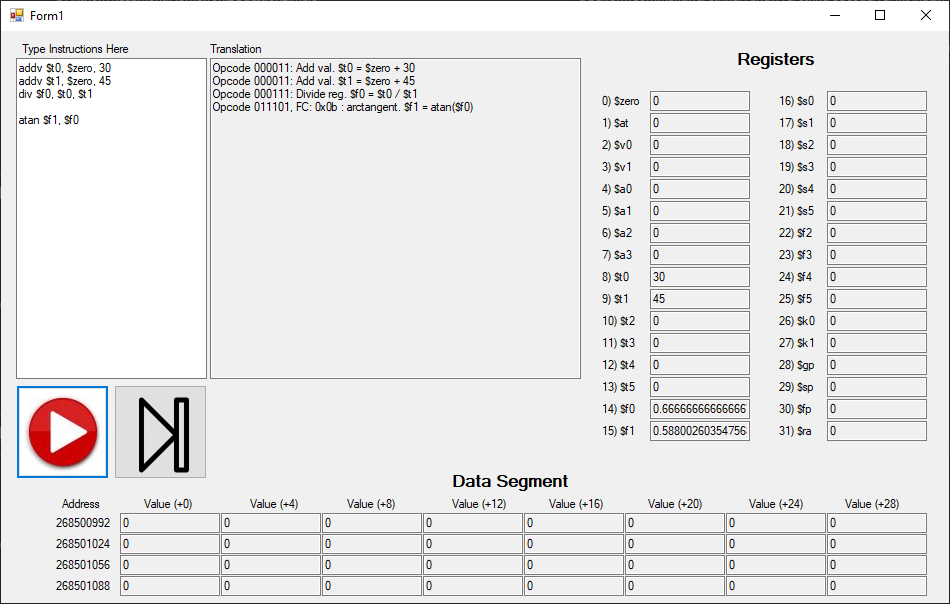


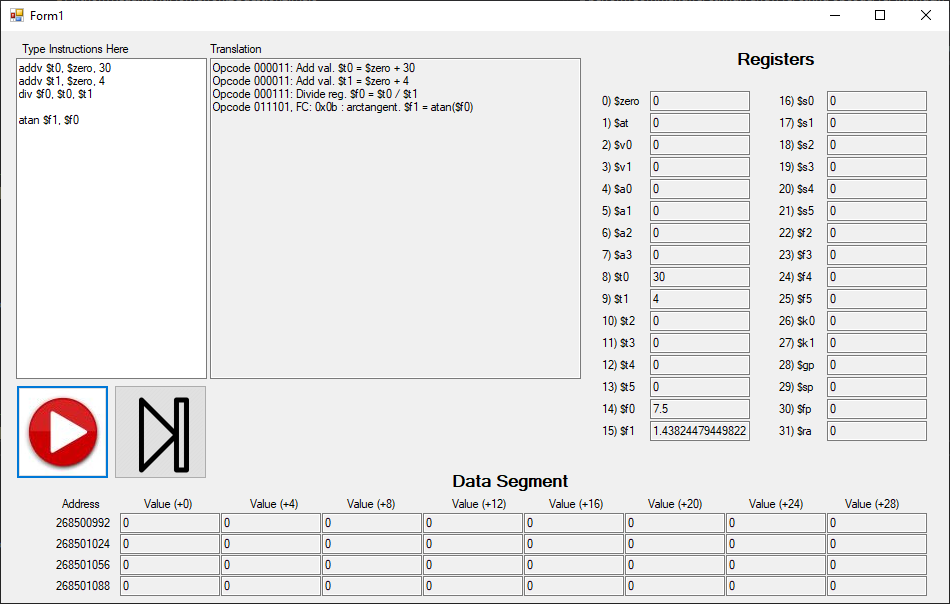


Atan: Opcode = 011101, Function Code = 0x0b

Rt is equal to the arctangent of Rs.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Bits** | 26-31 | 21-25 | 16-20 | 8-15 | 0-7 |
| **Meaning** | Opcode | Rs | Rt | 0x00 | Function Code |

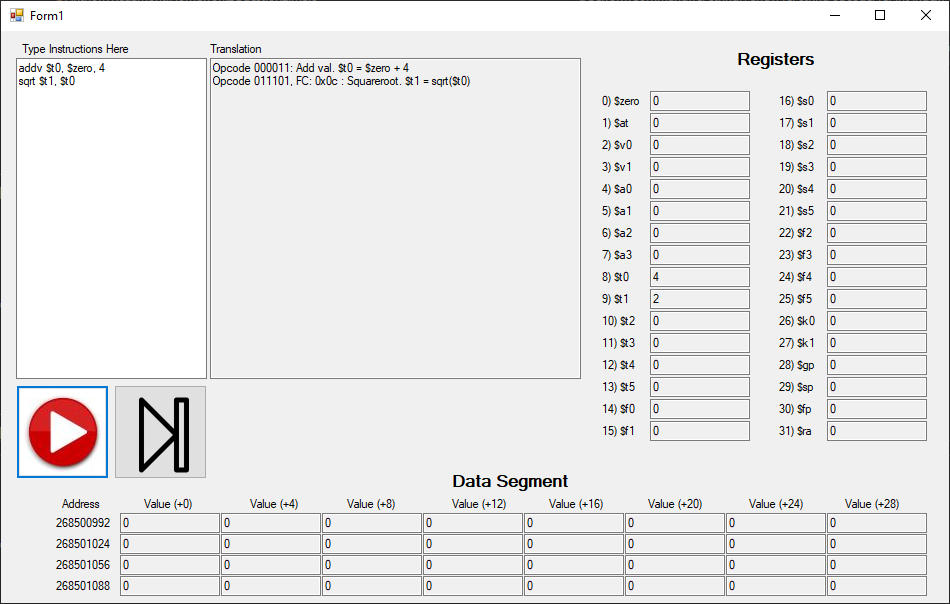


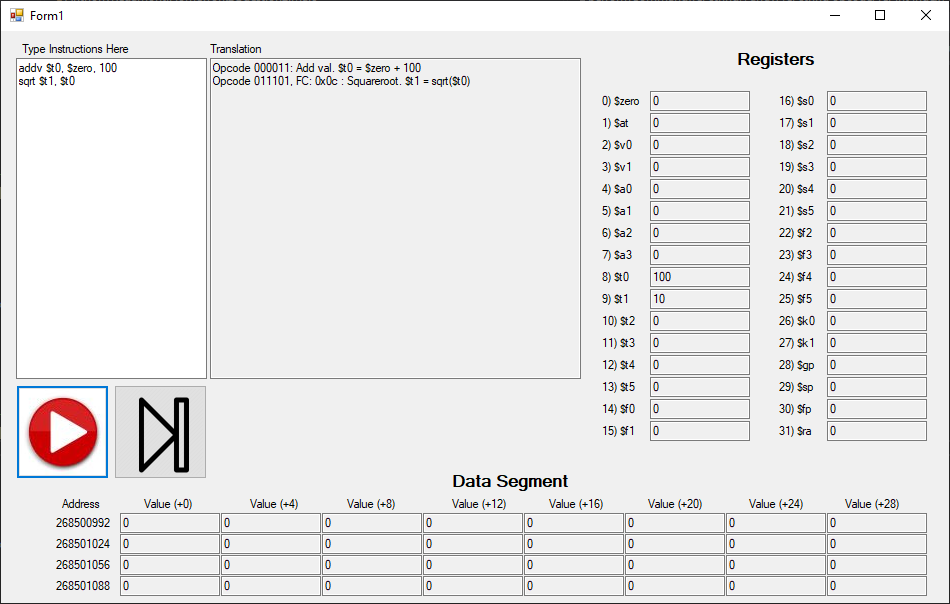


Sqrt: Opcode = 011101, Function Code = 0x0c

Rt is equal to the square root of Rs.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Bits** | 26-31 | 21-25 | 16-20 | 8-15 | 0-7 |
| **Meaning** | Opcode | Rs | Rt | 0x00 | Function Code |

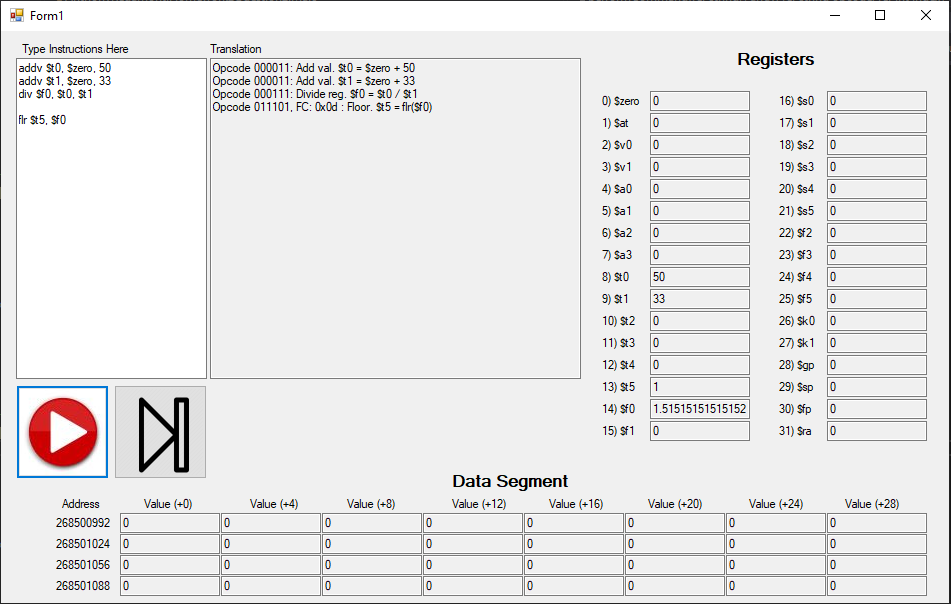


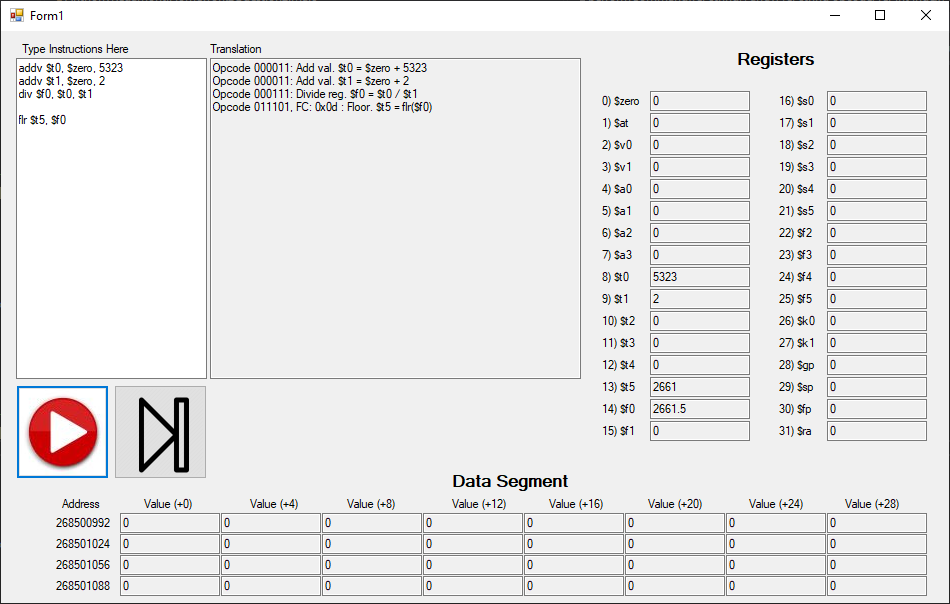


Flr: Opcode = 011101, Function Code = 0x0d

Rt is equal to the floor of Rs.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Bits** | 26-31 | 21-25 | 16-20 | 8-15 | 0-7 |
| **Meaning** | Opcode | Rs | Rt | 0x00 | Function Code |

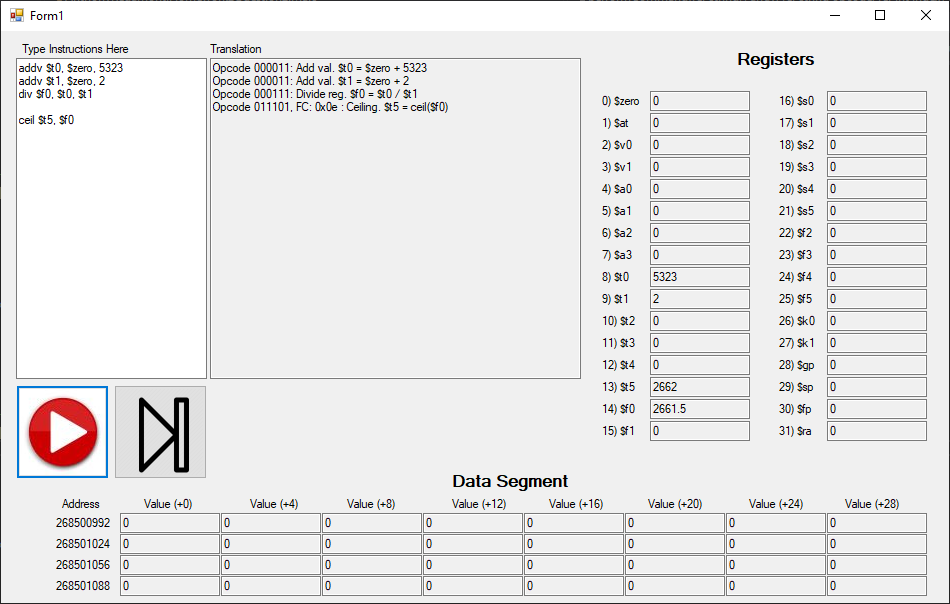


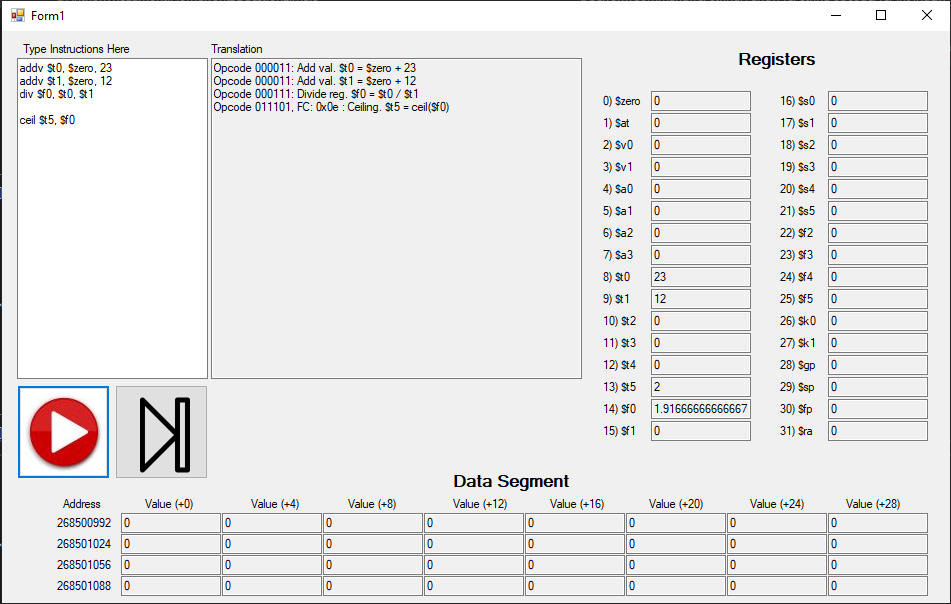


Ceil: Opcode = 011101, Function Code = 0x0e

Rt is equal to the ceiling of Rs.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Bits** | 26-31 | 21-25 | 16-20 | 8-15 | 0-7 |
| **Meaning** | Opcode | Rs | Rt | 0x00 | Function Code |

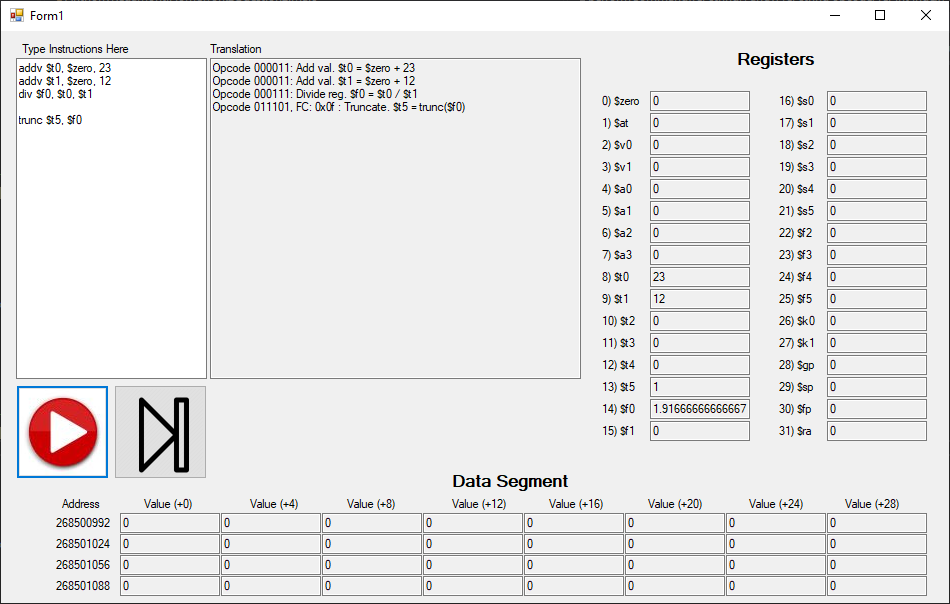


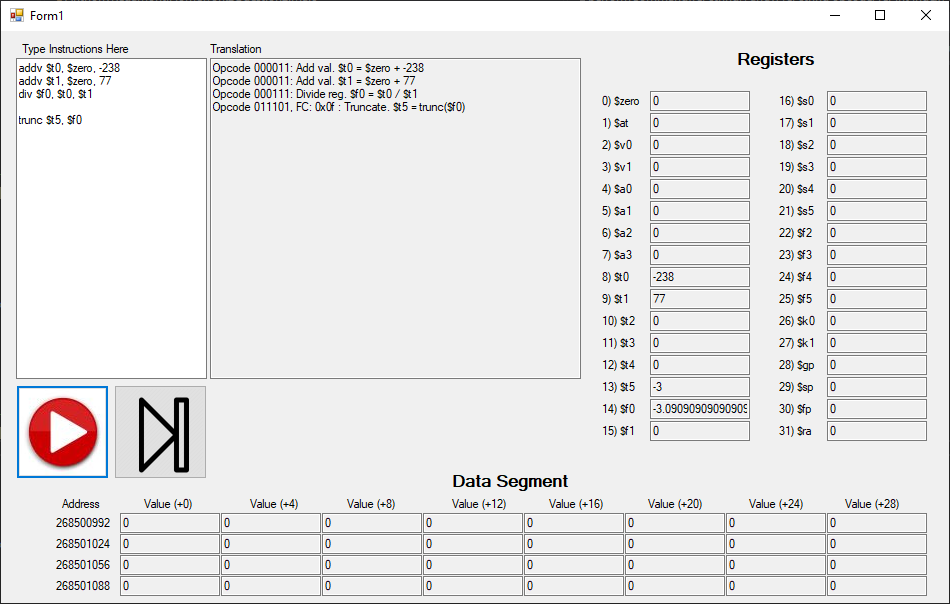


Trunc: Opcode = 011101, Function Code = 0x0f

Rt is equal to the truncate of Rs.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Bits** | 26-31 | 21-25 | 16-20 | 8-15 | 0-7 |
| **Meaning** | Opcode | Rs | Rt | 0x00 | Function Code |

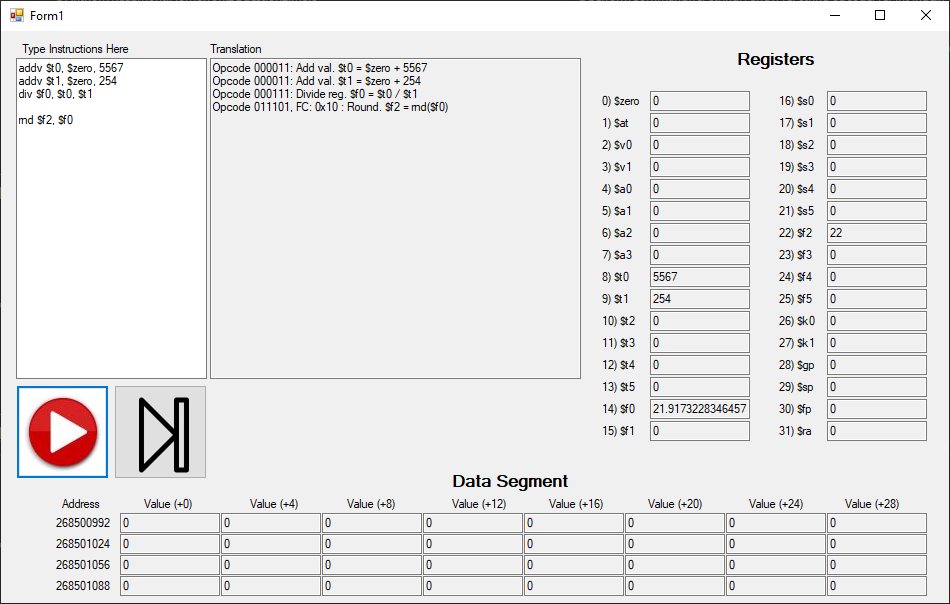


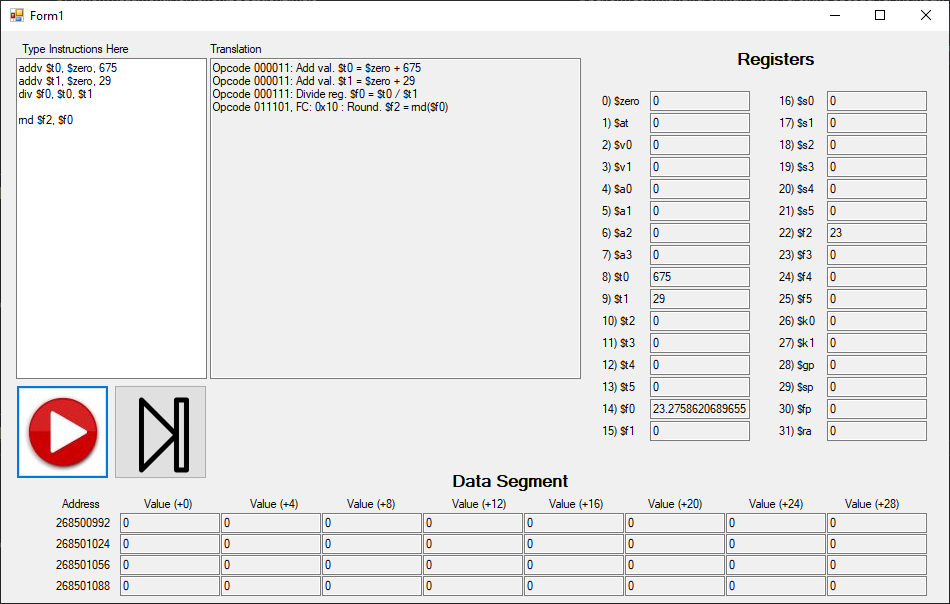


Rnd: Opcode = 011101, Function Code = 0x10

Rt is equal to the rounded value of Rs.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Bits** | 26-31 | 21-25 | 16-20 | 8-15 | 0-7 |
| **Meaning** | Opcode | Rs | Rt | 0x00 | Function Code |

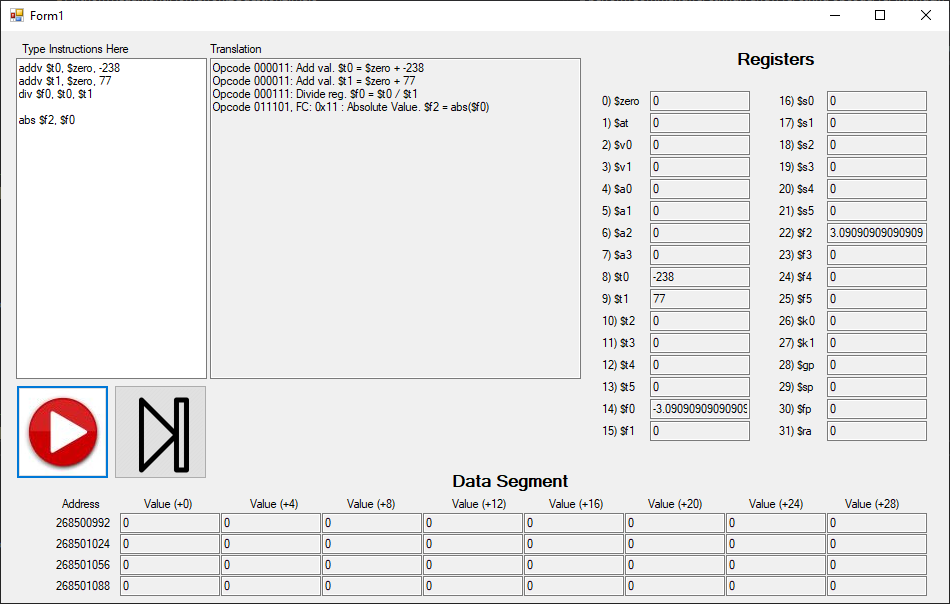


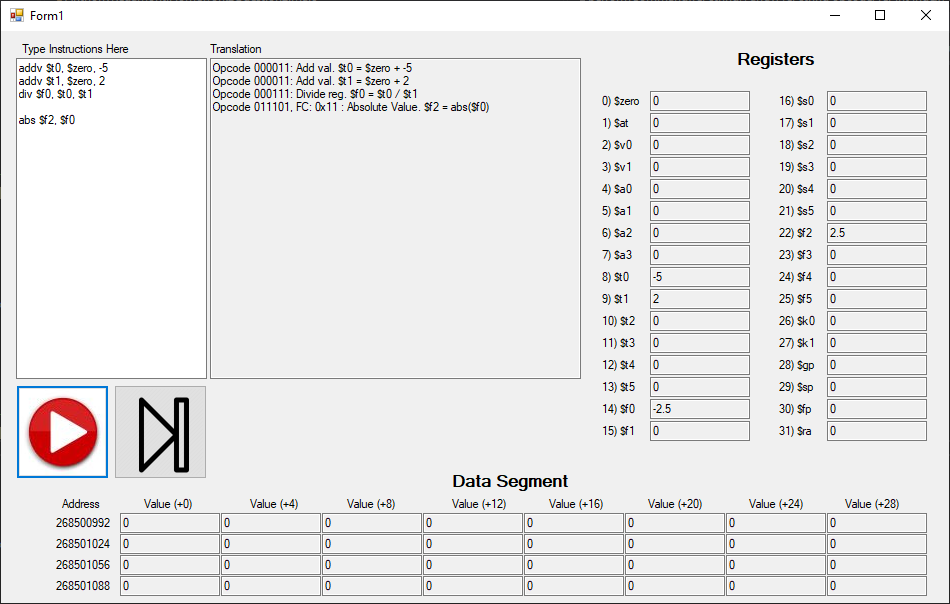


Abs: Opcode = 011101, Function Code = 0x11

Rt is equal to the absolute value of Rs.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Bits** | 26-31 | 21-25 | 16-20 | 8-15 | 0-7 |
| **Meaning** | Opcode | Rs | Rt | 0x00 | Function Code |

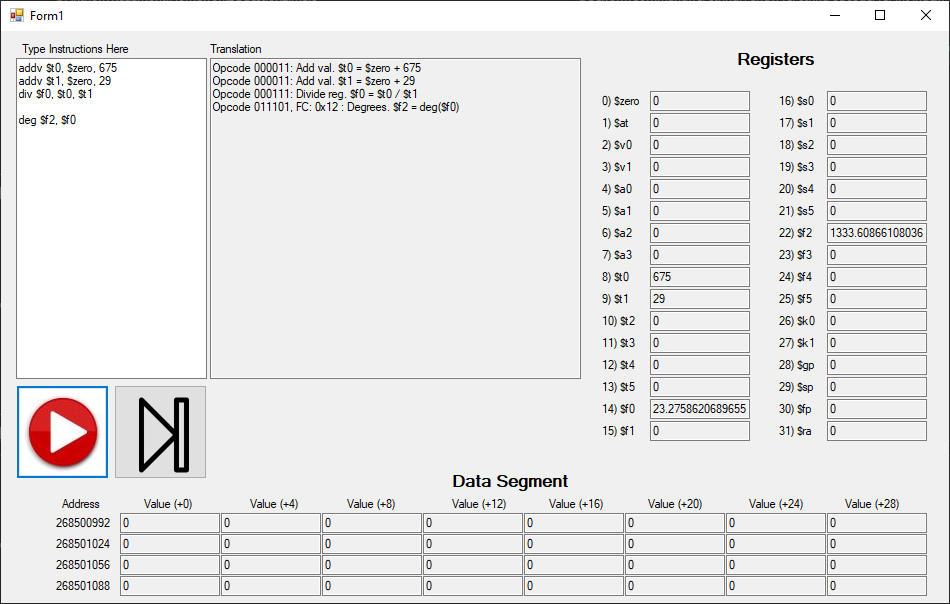


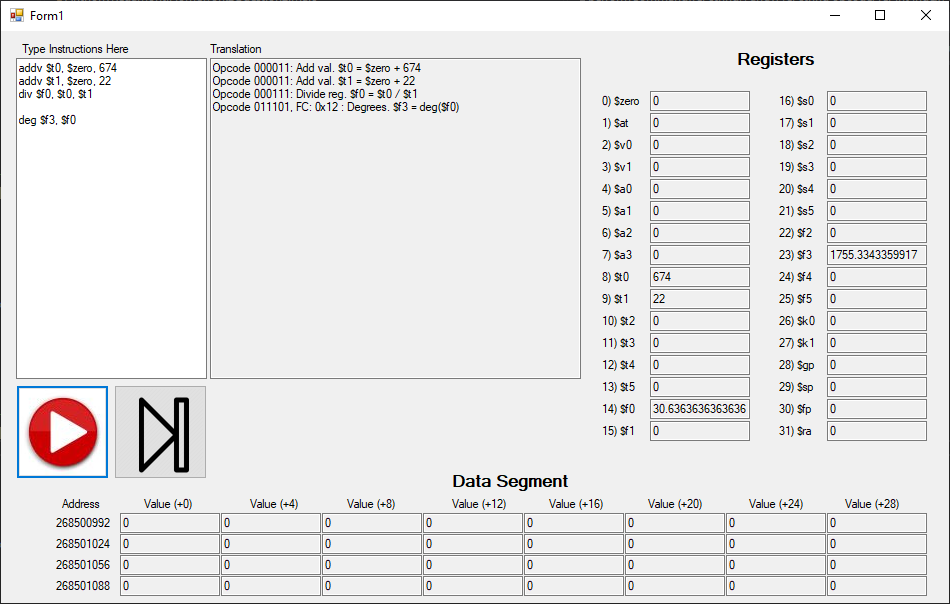


Deg: Opcode = 011101, Function Code = 0x12

Rs is in radians. Rt equals the conversion of Rs to degrees.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Bits** | 26-31 | 21-25 | 16-20 | 8-15 | 0-7 |
| **Meaning** | Opcode | Rs | Rt | 0x00 | Function Code |

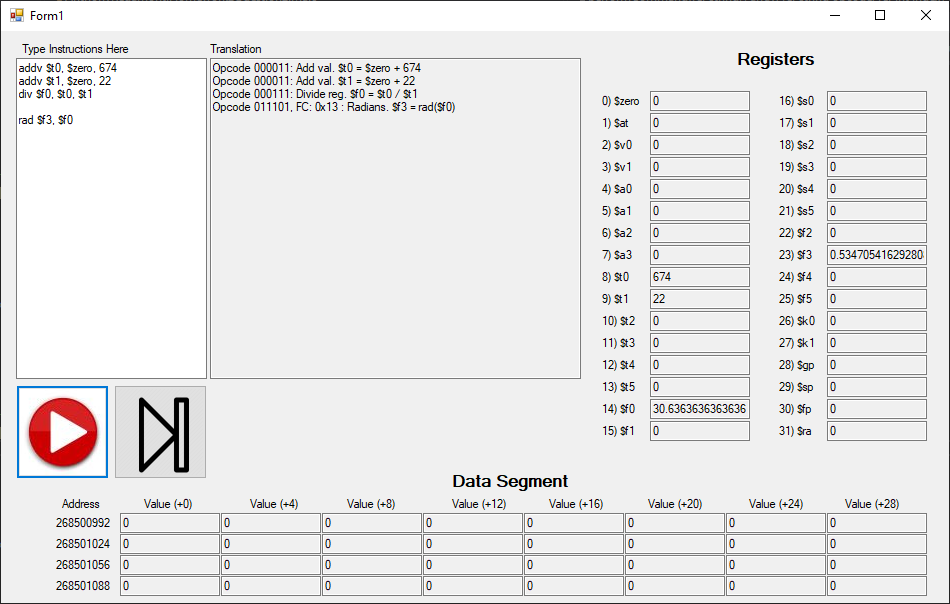


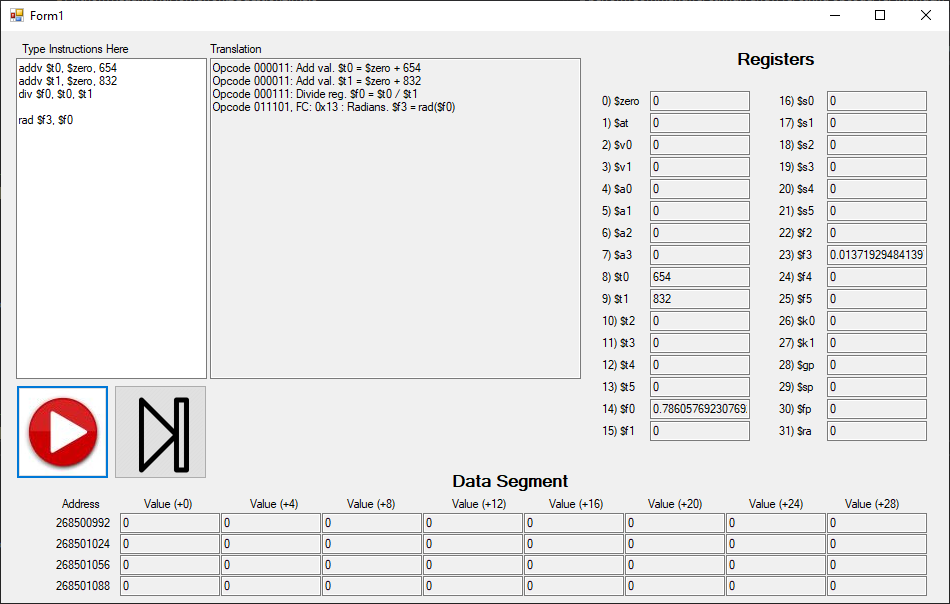


Rad: Opcode = 011101, Function Code = 0x13

Rs is in degrees. Rt equals the conversion of Rs to radians.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Bits** | 26-31 | 21-25 | 16-20 | 8-15 | 0-7 |
| **Meaning** | Opcode | Rs | Rt | 0x00 | Function Code |





Fact: Opcode = 011101, Function Code = 0x14

Rt equals the factorial of Rs.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Bits** | 26-31 | 21-25 | 16-20 | 8-15 | 0-7 |
| **Meaning** | Opcode | Rs | Rt | 0x00 | Function Code |

