

Special Instruction: cout

Formatted output for the LearnRisc simulator

What cout does

cout sends program output to the simulator's Run display. It can combine literal text, register values, hexadecimal values, and zero-terminated strings stored in byte memory. This lets a RISC-V program explain its results instead of requiring the user to inspect registers and memory manually.

Important: cout is a LearnRisc teaching extension. It is not a standard RV32I machine instruction. During assembly, each cout command is placed in the I/O definition table and represented in the executable instruction stream by a special simulator command.

General form

```
cout << item << item << endl;
```

The insertion operator << joins output items from left to right. A semicolon ends the command. Use ordinary straight quotation marks in source code.

Supported output items

Item	Purpose	Example
"text"	Print literal text exactly as written.	"Result = "
x0...x31	Print a register as a signed decimal value.	x21
ABI register	Print a register using an ABI name supported by LearnRisc.	s6 or a5
_to_hex register	Print the register's 32-bit value in hexadecimal.	_to_hex s6
_string register	Treat the register as a byte-memory address and print through the first zero byte.	_string a5
endl	Finish the current output line and begin a new line.	endl

Using cout in a program

Examples, memory strings, and simulator behavior

Common examples

Print literal text

```
cout << "Hello, world!" << endl;
```

Print a decimal register value

```
cout << "The resulting value = " << x21 << endl;
```

Print a register in hexadecimal

```
cout << "The resulting value = " << _to_hex s6 << endl;
```

Print a zero-terminated memory string

```
cout << "The message is: " << _string a5 << endl;
```

For `_string a5`, register `a5` must contain a byte address. The simulator reads one byte at a time, converts each nonzero byte to a character, and stops when it finds `0x00`. The zero terminator is not displayed.

Memory-string example

```
lui a5, 1 // a5 = 0x1000
addi x5, x0, 72 // ASCII 'H'
sb x5, 0(a5)
addi x5, x0, 105 // ASCII 'i'
sb x5, 1(a5)
addi x5, x0, 33 // ASCII '!'
sb x5, 2(a5)
sb x0, 3(a5) // zero terminator
cout << "Message: " << _string a5 << endl;
```

How LearnRisc processes cout

Source command	Assembler	Simulator
cout expression	Stores the complete command in the I/O definition table and emits its indexed special instruction.	Recognizes the special instruction, retrieves the indexed cout definition, evaluates its items, and appends text to the Run display.

Useful checks: Keep quoted text inside straight quotation marks; separate each item with `<<`; place `_to_hex` or `_string` immediately before its register; ensure every memory string ends with `0x00`; and avoid adding `endl` when the memory string already contains a newline unless you want an extra blank line.