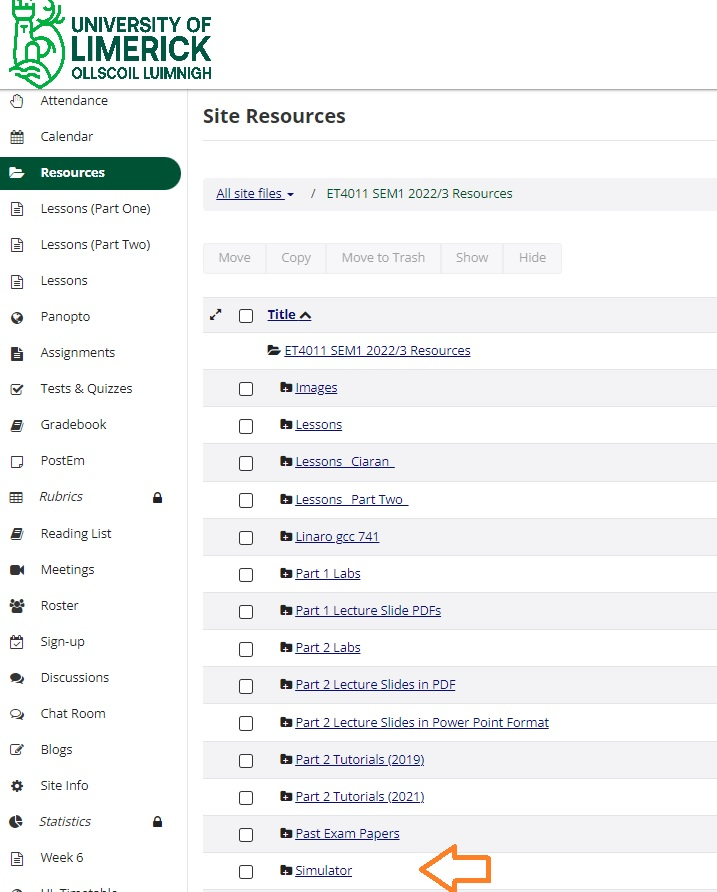
# Lab 7: LEGv8 Simulator

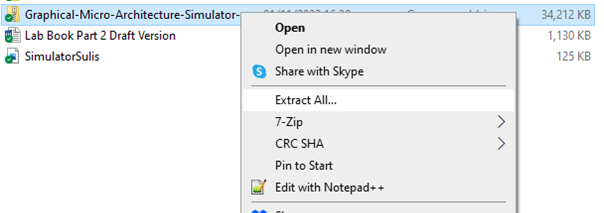
Version 1.2, Typos corrected and NZVC table entries simplified

**Getting Started**

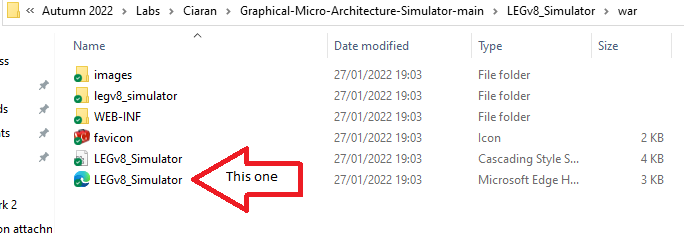
The LEGv8 Web-based simulator allows us to write and simulate simple programs that run on the LEGv8 microprocessor. LEGv8 is a subset of ARMv8, and contains all the instructions we will use in this semester. You can find the ZIP file for the simulator on Sulis.



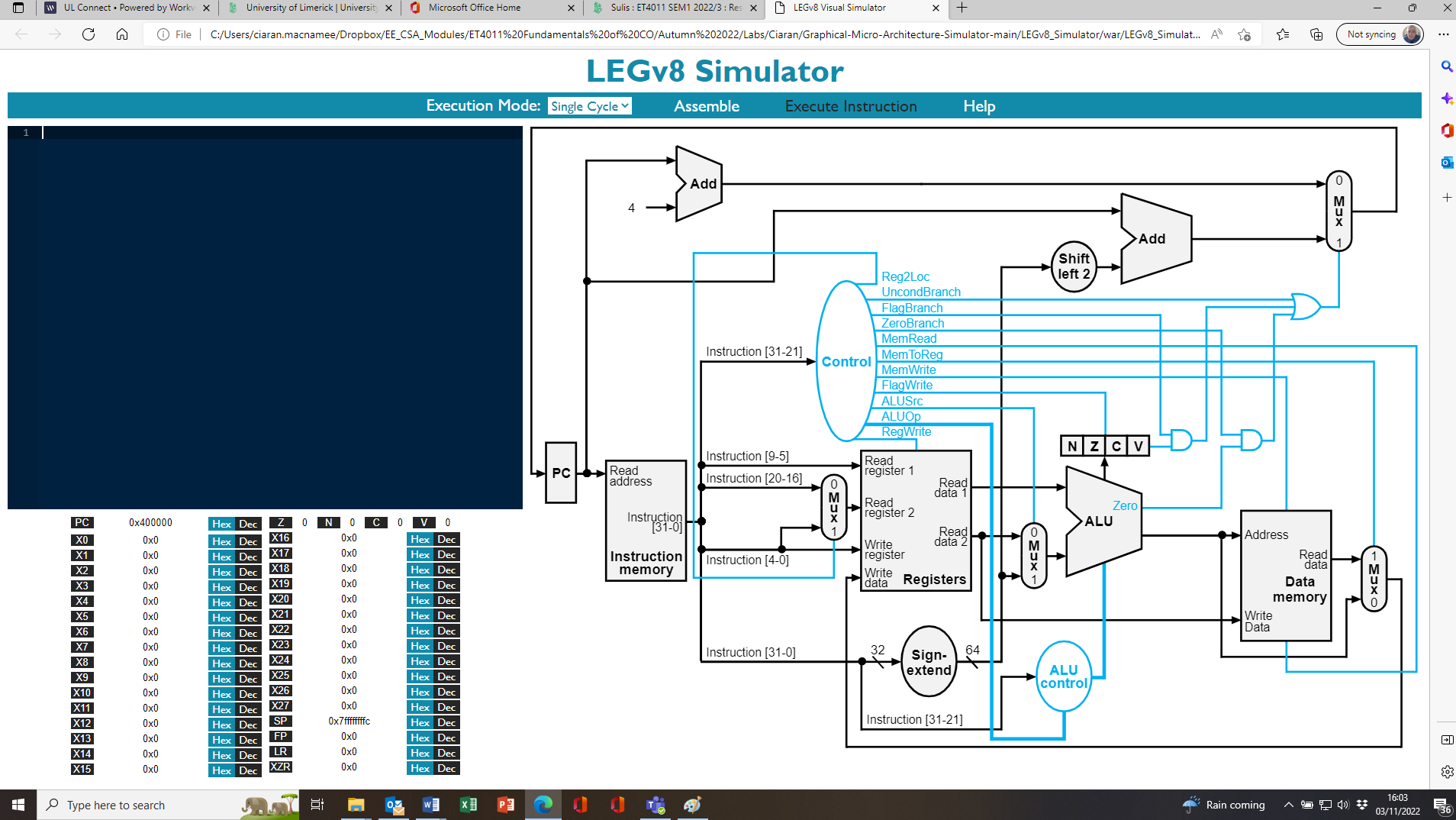
Download (**don’t open**) the file to your local computer and unzip or Extract All the file.



When you have extracted the file, browse through the folders you have created until you find the ‘war’ folder and click on the LEGv8\_Simulator HTML file (you may see a different icon):



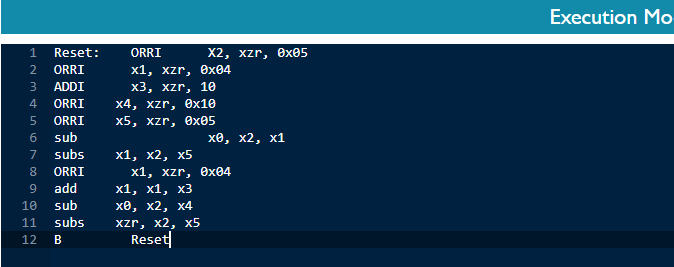
This opens up the LEGv8 Simulator



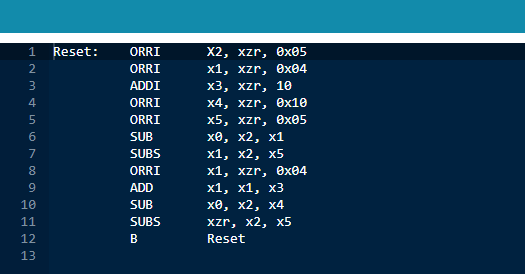
**Exercise 1**

By default LEGv8 simulated programs start at address 0x400000.

Download the file Exercise1.asm from Sulis and open it in a text editor such as Notepad. Use CTRL-A and CTRL-C to copy all the contents of the file. Place your cursor at the left top corner of the Black Box in screen. Use CTRL-V to paste the assembly source code you have just copied into the simulator code window.



Now Press ‘Assemble’. You should get no errors and the code will also line up nicely.



**Initial State:** What are the initial values of the following registers (before executing any instructions? (If you do not know the decimal values you may omit them).

|  |  |  |
| --- | --- | --- |
| Register Name | Value in Hexadecimal | Value in Decimal (If you know it) |
| PC | 0x400000 | 4194304 |
| X0 | 0x0 | 0 |
| X1 | 0x0 | 0 |
| X2 | 0x0 | 0 |
| X3 | 0x0 | 0 |
| X4 | 0x0 | 0 |
| X5 | 0x0 | 0 |
| XZR | 0x0 | 0 |

Now Press **Execute Instruction** to simulate your program execution, one instruction at a time and use the information on the simulator screen to fill in the blank spaces and answer the questions below.

**After ORRI X2, xzr, 0x05 Execution:**

|  |  |  |
| --- | --- | --- |
| Register Name | Value in Hexadecimal | Value in Decimal (If you know it) |
| PC | 0x400004 | 4194308 |
| X0 | 0x0 | 0 |
| X1 | 0x0 | 0 |
| X2 | 0x5 | 5 |
| X3 | 0x0 | 0 |
| X4 | 0x0 | 0 |
| X5 | 0x0 | 0 |
| XZR | 0x0 | 0 |

**After ORRI X1, xzr, 0x04 Execution:**

|  |  |  |
| --- | --- | --- |
| Register Name | Value in Hexadecimal | Value in Decimal (If you know it) |
| PC | 0x400008 | 4194312 |
| X0 | 0x0 | 0 |
| X1 | 0x4 | 4 |
| X2 | 0x5 | 5 |
| X3 | 0x0 | 0 |
| X4 | 0x0 | 0 |
| X5 | 0x0 | 0 |
| XZR | 0x0 | 0 |

**Q: What is the effect of the ORRI X1, xzr, 0x04 instruction?**

|  |
| --- |
| **Values of PC, x3 have changed** |

**After ADDI x3, xzr, 10 Execution:**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Register Name | Value in Hexadecimal | Z | N | C | V |
| PC | 0x40000c | 0 | 0 | 0 | 0 |
| X0 | 0x0 |  | | | |
| X1 | 0x4 |
| X2 | 0x5 |
| X3 | 0xa |
| X4 | 0x0 |
| X5 | 0x0 |
| XZR | 0x0 |

**After ORRI x4, xzr, 0x10 Execution:**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Register Name | Value in Hexadecimal | Z | N | C | V |
| PC | 0x400010 | 0 | 0 | 0 | 0 |
| X0 | 0x0 |  | | | |
| X1 | 0x4 |
| X2 | 0x5 |
| X3 | 0xa |
| X4 | 0x10 |
| X5 | 0x0 |
| XZR | 0x0 |

**Q: What are the bits Z, N, C and V used for?**

|  |
| --- |
| **Zero, Negative, Counter, Overflow** |

**After ORRI x5, xzr, 0x05 Execution:**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Register Name | Value in Hexadecimal | Z | N | C | V |
| PC | 0x400014 | 0 | 0 | 0 | 0 |
| X0 | 0x0 |  | | | |
| X1 | 0x4 |
| X2 | 0x5 |
| X3 | 0xa |
| X4 | 0x10 |
| X5 | 0x5 |
| XZR | 0x0 |

**After sub x0, x2, x1 Execution:**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Register Name | Value in Hexadecimal | Z | N | C | V |
| PC | 0x400018 | 0 | 0 | 0 | 0 |
| X0 | 0x1 |  | | | |
| X1 | 0x4 |
| X2 | 0x5 |
| X3 | 0xa |
| X4 | 0x10 |
| X5 | 0x5 |
| XZR | 0x0 |

**After subs x1, x2, x5 Execution:**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Register Name | Value in Hexadecimal | Z | N | C | V |
| PC | 0x40001c | 0 | 0 | 0 | 0 |
| X0 | 0x1 |  | | | |
| X1 | 0x0 |
| X2 | 0x5 |
| X3 | 0xa |
| X4 | 0x10 |
| X5 | 0x5 |
| XZR | 0x0 |

**After ORRI x1, xzr, 0x04 Execution:**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Register Name | Value in Hexadecimal | Z | N | C | V |
| PC | 0x400020 | 1 | 0 | 0 | 0 |
| X0 | 0x1 |  | | | |
| X1 | 0x4 |
| X2 | 0x5 |
| X3 | 0xa |
| X4 | 0x10 |
| X5 | 0x5 |
| XZR | 0x0 |

**After add x1, x1, x3 Execution:**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Register Name | Value in Hexadecimal | Z | N | C | V |
| PC | 0x400024 | 1 | 0 | 0 | 0 |
| X0 | 0x1 |  | | | |
| X1 | 0xe |
| X2 | 0x5 |
| X3 | 0xa |
| X4 | 0x10 |
| X5 | 0x5 |
| XZR | 0x0 |

**After sub x0, x2, x4 Execution:**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Register Name | Value in Hexadecimal | Z | N | C | V |
| PC | 0x400028 | 1 | 0 | 0 | 0 |
| X0 | 0xfffffffffffffff5 |  | | | |
| X1 | 0xe |
| X2 | 0x5 |
| X3 | 0xa |
| X4 | 0x10 |
| X5 | 0x5 |
| XZR | 0x0 |

**After subs xzr, x2, x5 Execution:**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Register Name | Value in Hexadecimal | Z | N | C | V |
| PC | 0x40002c | 1 | 0 | 0 | 0 |
| X0 | 0xe |  | | | |
| X1 | 0x5 |
| X2 | 0xa |
| X3 | 0x10 |
| X4 | 0x5 |
| X5 | 0x5 |
| XZR | 0x0 |

**After B Reset Execution:**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Register Name | Value in Hexadecimal | Z | N | C | V |
| PC | 0x400000 | 1 | 0 | 0 | 0 |
| X0 | 0xfffffffffffffff5 |  | | | |
| X1 | 0xe |
| X2 | 0x5 |
| X3 | 0xa |
| X4 | 0x10 |
| X5 | 0x5 |
| XZR | 0x0 |

**After ORRI X2, xzr, 0x05 Execution:**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Register Name | Value in Hexadecimal | Z | N | C | V |
| PC | 0x400004 | 1 | 0 | 0 | 0 |
| X0 | 0xfffffffffffffff5 |  | | | |
| X1 | 0xe |
| X2 | 0x5 |
| X3 | 0xa |
| X4 | 0x10 |
| X5 | 0x5 |
| XZR | 0x0 |

**Q: What is the effect of the B Reset Instruction?**

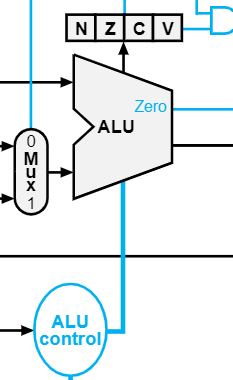
|  |
| --- |
| **Resets values and doesn’t read registers** |

**Q: What are the OPCODES for the following instructions?**

|  |  |
| --- | --- |
| ORRI x2, xzr, 0x05 | 1011001000 |
| ORRI x4, xzr, 0x10 | 1011001000 |
| ADDI x3, xzr, 10 | 10010000100 |
| add x1, x1, x3 | 10001011000 |
| sub x0, x2, x4 | 11001011000 |

**Advanced:**

Consider the following block of the CPU:

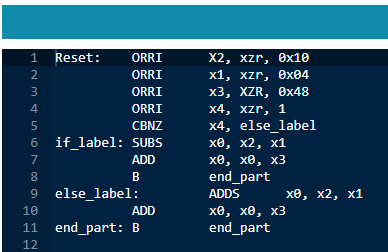


An identifier appears at the output of ALU control after a logic operation. What is the ALU Control value for the following instructions?

|  |  |
| --- | --- |
| ORRI | 0001 |
| ADDI | 0010 |
| ADD | 0010 |
| SUB | 0110 |
| SUBS | 0110 |

**Exercise 2**

Download the file Exercise2.asm from Sulis and open it in a text editor such as Notepad. Use CTRL-A and CTRL-C to copy all the contents of the file. Place your cursor at the left top corner of the Black Box in screen. Use CTRL-V to paste the assembly source code you have just copied into the simulator code window. Then Assemble the code as in Exercise 1.



Execute the code until you reach end\_part. What is the data contained in register X0?

|  |  |  |
| --- | --- | --- |
| Register Name | Value in Hexadecimal | Value in Decimal (If you know it) |
| X0 | 0x5c | 92 |

Modify the line ORRI x4, xzr, 1 to ORRI x4, xzr, 0

Assemble the code and repeat the execution to end\_part. What is the data contained in register X0?

|  |  |  |
| --- | --- | --- |
| Register Name | Value in Hexadecimal | Value in Decimal (If you know it) |
| X0 | 0x54 | 84 |

Suppose that x2 is variable a, x1 is variable b, x3 is variable c, x4 is variable Ctrl and x0 is variable y.

Write a number of lines of code in a pseudo high level language like Java or C that describes what is happening here. (Hint, it requires if…else).

A = 10

B = 4

C = 48

Ctrl = 1

If(Ctrl==0)

{

Y = A – B+C

}

Else{

Y = A+B+C

}

**Exercise 3 (Advanced)**

Modify the pseudocode of Exercise 2 so that instead of testing for equal to 0 or not equal to 0, you are testing for ‘greater than’. Write an ARMv8 (or LEGv8) assembly program based on exercise 2 to implement your revised pseudocode. Upload your code to Sulis.

(Hint: you will need a combination of SUBS/CMP and BRGT or BRLE instead of CBNZ).

A = 10

B = 4

C = 48

Ctrl = 1

If(Ctrl>0)

{

Y = A – B+C

}

Else{

Y = A+B+C

}

Reset: ORRI X2, xzr, 0x10

ORRI x1, xzr, 0x04

ORRI x3, XZR, 0x48

ORRI x4, xzr, 2

CMP x4, xzr

B.LE else\_label

if\_label: SUBS x0, x2, x1

ADD x0, x0, x3

B end\_part

else\_label: ADDS x0, x2, x1

ADD x0, x0, x3

end\_part: B end\_part