### The Course Component Library Guide

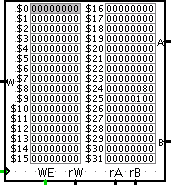
The Course Component Library, e.g, EEL4713.jar, is available on the course web page.

The md5sum is of EEL4713.jar is 7e69817da70f7930968add480578fb0c .

Please ensure that you are using the correct version.

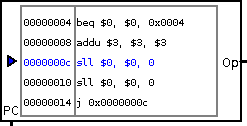
**Loading** the EEL4713 library From *Project* > *Load Library* > *Jar Library...* select the EEL4713.jar file. Also, put EEL4713.jar in the same folder with the .circ file. You should now have a new folder in your sidebar containing the following components:

**Register File.** A 32-bit wide by 32-registers deep register file. Register $0 is hard-wired to zero at all times, and writes to $0 are ignored. Inputs rA and rB are used to select register values to output on the A and B outputs. When the clock is triggered, if WE is high, the data value at input W is stored in register rW. The register file can be configured to use rising clock edges as trigger (the default), falling edge, or to be level sensitive.



**MIPS RAM.** This component is identical to Logisim's regular 32-bit wide, word-addressed RAM component, except the single-bit *sel* input is replaced by a 4-bit input to selectively enable or disable access to individual bytes within a 32-bit word. For instance, if the memory contains the word 0xAABBCCDD at the current address, then *sel*=0011 will access only the least significant halfword (0xCCDD), and *sel*=1000 will access only the most significant byte (0xAA). This component does not have a defined endianness since it is word addressed. Byte addressed memory (either little-endian or big-endian) can be emulated: to access byte-address 4 (the fifth byte in memory) in little-endian mode, one would set *a*=1 (second word) and *sel*=0001 (least significant byte); to access byte-address 4 (the fifth byte in memory) in big-endian mode, one would set *a*=1 (second word) and *sel*=1000 (most significant byte).

**MIPS Program ROM.** A 32-bit wide byte-addressed ROM, with built-in MIPS assembler. Use the attributes panel to load a MIPS assembly file into the ROM. The PC input specifies the address of the current instruction, and must be a multiple of 4. The output is the 32-bit machine code instruction at the PC address, or an error if the PC is invalid. Reading addresses outside the range of code supplied by the assembly file will cause the ROM to output zero, which happens to encode a no-op in MIPS. The assembly language is in the format described in the notes.



**MIPS ALU.** Computes a result as follows:

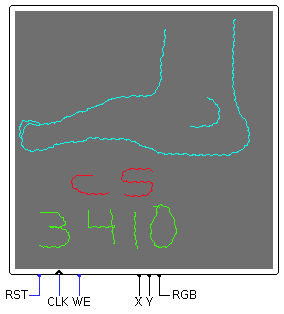
|  |  |  |
| --- | --- | --- |
| Op | name | C |
| ---- | ---------------------- | -------------- |
| 000x | shift left | C = B << Sa; |
| 001x | add | C = A + B |
| 0100 | shift right logical | C = B >>> Sa |
| 0101 | shift right arithmetic | C = B >> Sa |
| 011x | subtract | C = A - B |
| 1000 | and | C = A & B |
| 1010 | or | C = A | B |
| 1100 | xor | C = A ^ B |
| 1110 | nor | C = ~(A | B) |
| 1001 | eq | C = (A == B) ? 1 : 0 |
| 1011 | ne | C = (A != B) ? 1 : 0 |
| 1101 | gtz | C = (A > 0) ? 1 : 0 |
| 1111 | lez | C = (A ≤ 0) ? 1 : 0 |



**Incrementer.** An adjustable-width incrementer. Takes its input A on the left and outputs A+1 on the right.

http://www.cs.cornell.edu/courses/cs3410/2012sp/project/images/incr.png

**LCD Video.** If WE is high on the rising edge of CLK, writes a pixel on the LCD screen at the location given by the 7-bit unsigned X and Y coordinates. The pixel color is specified by 16-bit RGB input (in 5-5-5 format). The RST input resets the LCD screen. With some cleverness, you can redirect memory writes to this device to let your program draw on the LCD screen. Logisim also comes with some interesting input and output devices which can be used similarly.



### MIPS (subset) Assembly Syntax

The MIPS Program ROM component has a built-in assembler that understands all of the instructions you will implement. The syntax is standard MIPS syntax. Labels are case sensitive, everything else ignores case. Anything following a pound ('#') is a comment. In project 1, you will only use a few of the instructions listed here.

The instruction syntax is the same as given in the MIPS standard (and different from the output of gcc and many other tools). Registers are written as $0, $1, ..., $31, and the destination register goes on the left, followed by source registers and immediate values on the right. Most integer arguments (immediates, shift amounts, jump targets) can be specified in hex (i.e. 0x12ab), in decimal (i.e. 1234 or -1234), a label, or the special constant PC. The assembler will replace PC with the address of the instruction itself. Most constants have some restrictions: jump destinations must have the same upper 4 bits as the PC+4, and must be a multiple of 4; branch destinations must be a multiple of 4 and fit in a signed 18 bit immediate; etc. As a special case, when a branch target is specified symbolically as a label, the assembler will automatically subtract the current PC value to obtain a signed offset.

By default, the first instruction will be placed at address 0, and subsequent instructions are placed at at addresses 4, 8, 12, etc.

**Assembler directives.** The Program ROM assembler understands two standard MIPS assembler directives, .text and .word, both of which take integer (hex or decimal) arguments. For example, .text 0x50000000 will direct the assembler place subsequent instructions starting at address 0x50000000. And .word 0x24030005 directs the assembler to use the value 0x24030005 as the next machine instruction, which happens to be the machine code for ADDIU $3, $0, 5.

Some examples of instructions are:

|  |  |
| --- | --- |
| Immediate Arithmetic | ADDIU $12, $0, PC |
| Register Arithmetic | ADDU $13, $0, $20 |
| Immediate Load | LUI $14, 0x123 |
| Shifts | SLL $13, $13, 2  SLLV $15, $14, $3 |
| Jumps | J 0x24  J my\_label  JR $5  JALR $31, $5  JALR $5 |
| Branches | BEQ $5, $6, -12  BEQ $5, $6, my\_loop\_top  BLEZ $9, 16  BLEZ $9, my\_loop\_done |
| Memory Load/Store | LW $12, -4($30)  SW $12, 0($30) |

### MIPS (subset) Opcode Summary (from the MIPS Handbook)

Items in white are required for project 1 and project 2. Make sure to decode all of them.

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Table 1: MIPS32 Encoding of the Opcode Field | | | | | | | | | |
| **opcode** | | *bits 28..26 →* | |  | | | | | |
| ↓ bits 31..29 | | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
| 000 | 001 | 010 | 011 | 100 | 101 | 110 | 111 |
| 0 | 000 | *SPECIAL δ* | *REGIMM δ* | J | JAL | BEQ | BNE | BLEZ | BGTZ |
| 1 | 001 | ADDI | ADDIU | SLTI | SLTIU | ANDI | ORI | XORI | LUI |
| 2 | 010 | *COP0 δ* | *COP1 δ* | *COP2 θδ* | *COP3 θδ* | *BEQL φ* | *BNEL φ* | *BLEZL φ* | *BGTZL Φ* |
| 3 | 011 | β | β | β | β | *SPECIAL2 δ* | *JALX ε* | ε | \* |
| 4 | 100 | LB | LH | LWL | LW | LBU | LHU | LWR | β |
| 5 | 101 | SB | SH | SWL | SW | β | β | SWR | CACHE |
| 6 | 110 | LL | LWC1 | LWC2 θ | PREF | β | LDC1 | LDC2 θ | β |
| 7 | 111 | SC | SWC1 | SWC2 θ | \* | β | SDC1 | SDC2 θ | β |

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Table 2: MIPS32 *SPECIAL* Opcode Encoding of the Function Field | | | | | | | | | |
| **function** | | *bits 2..0 →* | |  | | | | | |
| ↓ bits 5..3 | | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
| 000 | 001 | 010 | 011 | 100 | 101 | 110 | 111 |
| 0 | 000 | SLL | *MOVCI δ* | SRL | SRA | SLLV | \* | SRLV | SRAV |
| 1 | 001 | JR | JALR | MOVZ | MOVN | SYSCALL | BREAK | \* | SYNC |
| 2 | 010 | MFHI | MTHI | MFLO | MTLO | β | \* | β | β |
| 3 | 011 | MULT | MULTU | DIV | DIVU | β | β | β | β |
| 4 | 100 | ADD | ADDU | SUB | SUBU | AND | OR | XOR | NOR |
| 5 | 101 | \* | \* | SLT | SLTU | β | β | β | β |
| 6 | 110 | TGE | TGEU | TLT | TLTU | TEQ | \* | TNE | \* |
| 7 | 111 | β | \* | β | β | β | \* | β | β |

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Table 3: MIPS32 *SPECIAL* REGIMM Encoding of the rt Field | | | | | | | | | |
| **rt** | | *bits 18..16 →* | |  | | | | | |
| ↓ bits 20..19 | | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
| 000 | 001 | 010 | 011 | 100 | 101 | 110 | 111 |
| 0 | 00 | BLTZ | BGEZ | BLTZL | BGEZL | \* | \* | \* | \* |
| 1 | 01 | TGEI | TGEIU | TLTI | TLTIU | TEQI | \* | TNEI | \* |
| 2 | 10 | BLTZAL | BGETAL | BLTZALL | BGETALL | \* | \* | \* | \* |
| 3 | 11 | \* | \* | \* | \* | \* | \* | \* | \* |