# EE20021 HighRisc Processor instruction set definition

C. T. Clarke

November 2015

**Introduction**

This document describes the instruction set of the HighRisc processor used in the tutorials and labs for EE20021 Digital Systems Design

**Instruction formats**

There are two formats for the instruction set as shown in table 1.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Section** | **Opcode** | | | | **Destination (Dest)** | | | | | | **Source (Src) / Immediate** | | | | | |
| Value | 1 to 15 | | | | 0 to 63 | | | | | | 0 to 63 | | | | | |
| Bit | 15 |  |  | 12 | 11 |  |  |  |  | 6 | 5 |  |  |  |  | 0 |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| **Section** | **Opcode** | | | | **Condition** | | | **Offset** | | | | | | | | |
| Value | 0 | | | | See Flags | | | -256 to 255 | | | | | | | | |
| Bit | 15 |  |  | 12 | 11 |  | 9 | 8 |  |  |  |  |  |  |  | 0 |

Table 1. Instruction formats

**Registers**

The system has 62 general purpose registers (R0 to R61). Each is 16 bits in size. There are also two special registers as described below

Flags

Register 62 is the Flags register (FL). This is a 5 bit register containing the ALU result flags described in table 2. Three conditions are also described in the table that are derived from the flags but not stored in the register. A minimal implementation provides a Carry flag the other flags are optional.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Bit** | **Condition** | **Code** | **Description** | **Essential** |
| 0 | Carry | C | Set when an arithmetic function produces a carry (or borrow). Also used as an extra register bit in rotate operations | Yes |
| 1 | Zero | Z | Set when the result of an ALU operation is zero |  |
| 2 | Negative | N | Set when the result of an ALU operation is negative |  |
| 3 | Parity | P | Set when the number of 1 bits in the result of and ALU operation is even (including when no bits are set to 1) |  |
| 4 | Overflow | V | Set when an ALU addition or subtraction operation overflows. The conditions in which this is deemed to have occurred are shown in table 3 |  |
| - | No carry | NC | Set when an arithmetic function doesn’t produce a carry (or borrow). Rotate operations affect this flag |  |
| - | Not zero | NZ | Set when the result of an ALU operation is not zero |  |
| - | Always | A | Always set |  |

Table 2. Flags and conditions

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Addition** | | | |  | **Subtraction** | | | |
| **Dest MSB** | **Src**  **MSB** | **Result**  **MSB** | **V flag** |  | **Dest MSB** | **Src**  **MSB** | **Result**  **MSB** | **V flag** |
| 0 | 0 | 0 | 0 |  | 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 1 |  | 0 | 0 | 1 | 0 |
| 0 | 1 | 0 | 0 |  | 0 | 1 | 0 | 0 |
| 0 | 1 | 1 | 0 |  | 0 | 1 | 1 | 1 |
| 1 | 0 | 0 | 0 |  | 1 | 0 | 0 | 1 |
| 1 | 0 | 1 | 0 |  | 1 | 0 | 1 | 0 |
| 1 | 1 | 0 | 1 |  | 1 | 1 | 0 | 0 |
| 1 | 1 | 1 | 0 |  | 1 | 1 | 1 | 0 |

Table 3. Overflow flag conditions based on the most significant bit (MSB) of each or the ALU inputs and outputs

Program Counter

Register 63 is the Program Counter (PC) which stores the address of the currently executing instruction.

**Instructions**

Table 4 shows the instructions are executed by the processor. A minimal implementation only provides the instructions listed as essential. **For non-ALU operations, the ALU should output 0 on the OutDest port.**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Op-code Value** | **Name** | **Description** | **Flags Affected** | **Esse-ntial?** | **ALU op?** |
| 0 | JR | Changes the PC by the offset if the condition flag is set |  |  | No |
| 1 | LOAD | Loads register Dest with the value at the address in data memory given by register Src. |  | Yes | No |
| 2 | STORE | Copies the value in register Src to the address in data memory given by register Dest. |  | Yes | No |
| 3 | MOVE | Copies the content of register Src to register Dest. |  |  | Yes |
| 4 | NAND | Sets register Dest to the bitwise logical NAND of the contents of registers Dest and Src. |  | Yes | Yes |
| 5 | NOR | Sets register Dest to the bitwise logical NOR of the contents of registers Dest and Src. |  |  | Yes |
| 6 | ROL | Sets register Dest to the contents of register Src having first shifted the value left by 1 bit and sets the least significant bit of Dest to the value in the C flag. Following this operation, the C flag contains the most significant bit of SRC. | C | Yes | Yes |
| 7 | ROR | Sets register Dest to the contents of register Src having first shifted the value right by 1 bit and sets the most significant bit of Dest to the value in the C flag. Following this operation, the C flag contains the least significant bit of SRC. | C |  | Yes |
| 8 | LIL | Sets the contents of register Dest to a sign extended copy of the immediate value. |  |  | Yes |
| 9 | LIU | Sets the upper bits of register Dest based upon the immediate value as shown in table 5. |  |  | Yes |
| 10 | ADC | Sets the value of register Dest to be the sum of Src Dest and the C flag. All flags are set according to the result. | CZNVP |  | Yes |
| 11 | SUB | Sets the value of register Dest to Dest – (Src + C) flag. All flags are set according to the result. | CZNVP |  | Yes |
| 12 | DIV | Sets the value of register Dest to the result of a signed integer division Dest / Src. | ZNP |  | Yes |
| 13 | MOD | Sets the value of register Dest to the remainder of the signed integer division Dest / Src. | ZNP |  | Yes |
| 14 | MUL | Sets the value of register Dest to the low half of the signed integer product Dest x Src. | ZNP |  | Yes |
| 15 | MUH | Sets the value of register Dest to the high half of the signed integer product Dest x Src. | ZNP |  | Yes |

Table 4. The instructions

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Opcode** | **Imm[5]** | **Dest** | | | | | | | | | | | | | | | |
| LIU | 1 | Imm[4:0] | | | | | Unchanged | | | | | | | | | | |
| LIU | 0 | Sign Extend | | | | | Imm[4:0] | | | | | Unchanged | | | | | |
| LIL | X | Sign Extend | | | | | | | | | | Imm | | | | | |
|  | Bit | 15 |  |  |  | 11 | 10 |  |  |  | 6 | 5 |  |  |  |  | 0 |

Table 5a. Load Immediate instructions – 16 bit data width, 6 bit immediate

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Opcode** | **Imm[1]** | **Dest** | | | |
| LIU | 1 | Imm[0] | Unchanged | | |
| LIU | 0 | Sign Extend | Imm[0] | Unchanged | |
| LIL | X | Sign Extend | | Imm | |
|  | Bit | 3 | 2 | 1 | 0 |

Table 5a. Load Immediate instructions – 4 bit data width, 2 bit immediate