**ERM16**

**ERM16 -** Easy RISC Microprocessor 16 bits.

Simple 16-bit microprocessor with simplified set of instructions.

**Bit-depth:** 16 bits

**word:** 16 bits

**instruction set type:** RISC

**processor design type:** multi-cycle

**instruction set qty:** 41

**PORTS:**

1. **input:** 
   * + **clk:** clock signal
     + **VDD:** logic 1
     + **GND:** logic 0
     + **DI:** data bus 16 bit
     + **init:** processor initialization signal (hardware reset)
2. **output:**
   * + **IOE:** signal that requests I/O devices to their memory
     + **INTREQ:** a signal to notify the bus controller that smth is going to interrupt controller / signal to request an interrupt from the processor
     + **addr\_bus:** addressable 16-bit bus
     + **wrmem:** memory write enable signal
     + **DO:** 16-bit data bus

**INSTRUCTION SET:**

* **rd** register destination
* **rs** register source
* **imm6** immediate value 6 bits

**COMMANDS:**

|  |  |  |
| --- | --- | --- |
| **Name of command** | **Explanation** | **Syntax** |
| **ldw (load word)** | load word from memory into register, addressing register only via any general register | **ldw rd,[r]** |
| **stw (store word)** | store 16-bit value from register to memory, addressing register only via any general register | **stw rs,[r]** |
| **mov (move)** | copy a value from one register to another, or load a direct 6-bit value into a register | **mov rd,rs/imm6** |
| **j (jump)** | unconditional jump | **j rd/addr9** |
| **jz (jump if zero)** | jump if ZF = 1 | **jz rd/addr9** |
| **jc (jump if carry)** | jump if CF = 1 | **jc rd/addr9** |
| **jo (jump if oveflow)** | jump if OF = 1 | **jo rd/addr9** |
| **jp (jump if parity)** | jump if PF = 1 | **jp rd/addr9** |
| **jnz (jump if no zero)** | jump if ZF = 0 | **jnz rd/addr9** |
| **jnc (jump no carry)** | jump if CF = 0 | **jnc rd/addr9** |
| **jno (jump no overflow)** | jump if OF = 0 | **jno rd/addr9** |
| **jnp (jump no parity)** | jump if PF = 0 | **jnp rd/addr9** |
| **jg (jump if greater)** | jump if GF = 1 | **jg rd/addr9** |
| **jl (jump if lower)** | jump if LF = 1 | **jl rd/addr9** |
| **jge (jump if greater or equal)** | jump if ZF = 1 or GF = 1 | **jge rd/addr9** |
| **jle (jump if lower or equal)** | jump if LF = 1 or ZF = 1 | **jle rd/addr9** |
| **ret (return)** | return from the function, the processor extracts the value from the RAF register and loads it into the PC | **ret** |
| **ch (call handler)** | saves to raf by increasing the value by 2 and loading jump address into PC | **ch rd/addr9** |
| **int (interrupt)** | hardware interrupt request to device following vector | **int vector8** |
| **rst (reset cpu)** | reset cpu to initialization state | **rst** |
| **hlt (halt cpu)** | stop cpu | **hlt** |
| **add (+)** | rd = rd + rs + ci | **add rd,rs.** |
| **sub (-)** | rd =(rd – rs) - ci | **sub rd,rs.** |
| **mul (\*)** | mul rd,rs. | **Rd = rd\*rs** |
| **div (/)** | div rd,rs. | **Rd = rd/rs** |
| **lsl (<<)** | rd = rd << rs/imm6 | **lsl rd,rs/imm6.** |
| **mod (%)** | rd  = rd % rs/imm6 | **mod rd,rs/imm6** |
| **asr (>>>)** | rd = rd >>> rs,imm4 | **asr rd,rs/imm6.** |
| **lsr (>>)** | rd = rd >> rs,imm4 | **lsr rd,rs/imm6.** |
| **or** | Rd = rd | rs/imm6 | **or rd,rs/imm6.** |
| **eor (exlusive or)** | Rd = rd ^ rs | **eor rd,rs/imm6.** |
| **orn (or not)** | Rd = rd ~| rs | **orn rd,rs/imm6.** |
| **and** | Rd = rd & rs | **and rd,rs/imm6.** |
| **andn (and not)** | rd = rd ~& rs | **and rd,rs/imm6.** |
| **eon (exlusive or not)** | rd = rd ~^ rs | **eon rd,rs/imm6 .** |
| **mvn (move not)** | rd = ~rs/imm8 | **mvn rd,rs/imm6.** |
| **rev (reverse bits)** | Swap bytes from swap high byte becomes low 0xadf1 -> 0xf1ad | **rev rd,rs** |
| **in (read input port I/O devices)** | read data from the I/O device buffer | **in rd,r/imm6** |
| **out (write output port I/O devices)** | write data to I/O device buffer | **out r/imm6,rs** |
| **cmp (compare)** | compare rd with rs or with imm6 | **cmp rd,rs/imm6** |

**COMMAND PREFIX OR EXTENSION:**

* **r – register** (indicates that the processor works only with processor registers)
* **i – immediate** (indicates that the processor works with both registers and immediate values)

**Prefixes are mandatory in all instructions except for ldw stw alr rst hlt int ch ret.**

**Example:**

movr r2,r5

movi r2,0x4f

instruction entry:

instruction prefix/extension operand1,operand2

[15:9] [8:6] [5:3]/[5:0]

Instruction size 16 bits

instr r/i op1,op2

**PROCESSOR REGISTERS:**

* **r0-r6 (register):** 7 general purpose registers
* **pc (program counter):** instruction counter stores the current instruction address
* **raf/r7 (return address function register):** The function return address register contains the return address

**FLAGS (REGISTER FLAGS)**

* **ZF** zero flag
* **CF** carry flag
* **OF** overflow flag
* **GF** flag “greater than”
* **LF** flag “less than”
* **PF** parity flag