**ERM16**

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

**a simple 16-bit microprocessor with a simplified set of instructions.**

**bit depth: 16 bit**

**word : 16 bits**

**instruction set type : RISC**

**processor design type: multi-cycle**

**instruction set qty: 41**

**ports:**

**input:**

**clk : clock signal**

**VDD : logic 1**

**GND: logic 0**

**DI: data bus 16 bit**

**init: processor initialization signal (hardware reset)**

**weekends:**

**IOE : a signal that requests I/O devices to their memory**

**INTREQ : a signal to notify the bus controller that something is going to the interrupt controller or a 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:**

**ldw (load word) ldw rd,[r] load word from memory into register, addressing register only via any general register**

**stw (store word) stw rs,[r] store 16 bit value from register to memory, addressing register only via any general register**

**mov (move) mov rd,rs/imm6 copy a value from one register to another, or load a direct 6-bit value into a register**

**j (jump) j rd/addr9 unconditional jump**

**jz (jump if zero) jz rd/addr9 jump if ZF = 1**

**jc (jump if carry) jc rd/addr9 jump if CF = 1**

**jo (jump if oveflow) jo rd/addr9 jump if OF = 1**

**jp (jump if parity) jp rd/addr9 jump if PF = 1**

**jnz (jump if no zero) jnz rd/addr9 jump if ZF = 0**

**jnc (jump no carry) jnc rd/addr9 jump if CF = 0**

**jno (jump no overflow) jno rd/addr9 jump if OF = 0**

**jnp (jump no parity) jnp ard/ddr9 conditional jump if PF = 0**

**jg (jump if greater) jg rd/addr9 jump if GF = 1**

**jl (jump if lower) jl rd/addr9 jump if LF = 1**

**jge (jump if greater or equal) jge rd/addr9 jump if ZF = 1 or GF = 1**

**jle (jump if lower or equal) jle rd/addr9 jump if LF = 1 or ZF = 1**

**ret (return) ret return from the function, the processor extracts the value from the RAF register and loads it into the PC**

**ch (call handler) ch rd/addr9 saves to raf by increasing the value by 2 and loading jump address into pc**

**int (interrupt) int vector8 hardware interrupt request to device following vector**

**rst (reset cpu) rst reset cpu to initialization state**

**hlt (halt cpu) hlt stop cpu**

**add (+) add rd,rs. rd = rd + rs + ci**

**sub (-) sub rd,rs. rd = (rd - rs) - ci**

**mul (\*) mulrd,rs. Rd = rd\*rs**

**div (/) divrd,rs. Rd = rd/rs**

**lsl (<<) lslrd,rs/imm6 . rd = rd << rs/imm6**

**mod (%) mod rd,rs/imm6 rd = rd % rs/imm6**

**asr (>>>) asrrd,rs/imm6. rd = rd >>> rs,imm4**

**lsr (>>) lsrrd,rs/imm6 . rd = rd >> rs,imm4**

**or - or rd,rs/imm6. Rd = rd | rs/imm6**

**eor (exlusive or) eor rd,rs/imm6. Rd=rd^rs**

**orn (or not) orn rd,rs/imm6. Rd = rd~| rs**

**and – and rd,rs/imm6. Rd = rd & rs**

**andn (and not) and rd,**