Процессорное ядро SchoolMIPS

и его использование для обучения основам микроархитектуры процессора

Станислав Жельнио
github.com/zhelnio
2017

Проектирование микросхем для школьников

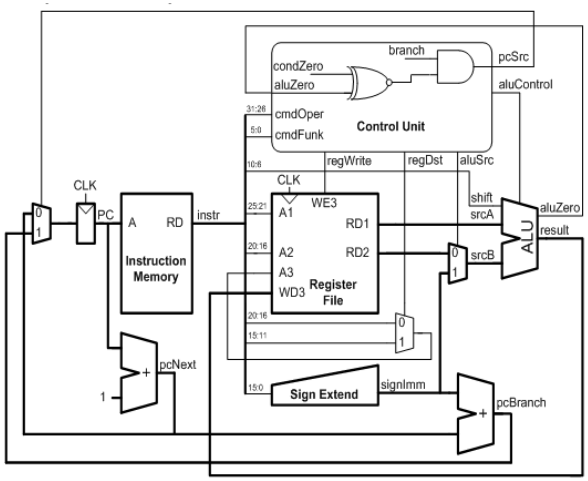
- **Теория**
  - Учебник «Цифровая схемотехника и архитектура компьютера»
    Дэвид Харрис, Сара Харрис (H&H)
- **Практика**
  - schoolMIPS
    простейший процессор
  - MIPSfpga
    промышленное процессорное ядро

Процессорное ядро schoolMIPS

- максимально упрощенное ядро архитектуры MIPS
  300 строк на языке Verilog
  основано на процессоре Сары Харрис (из H&H)

- детальная документация на русском языке
  слайды + руководство пользователя

- открытый исходный код
  https://github.com/MIPSfpga/schoolMIPS

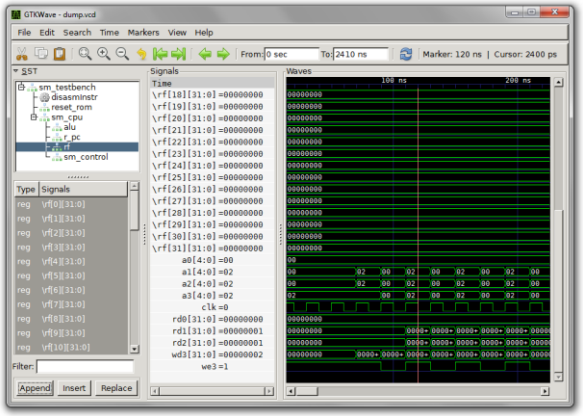


- предварительные требования:
  - основы языка описания аппаратуры Verilog
    на котором написан процессор

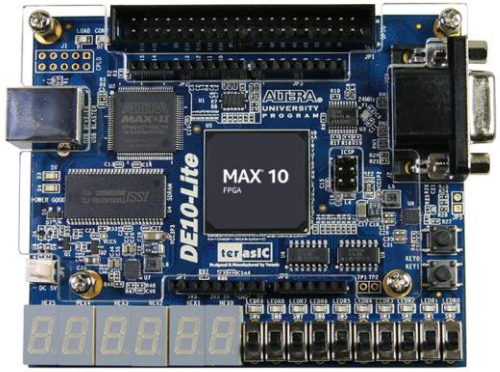
  - основы программирования на ассемблере MIPS
    сборка программ для процессора выполняется «взрослым» компилятором gcc

Что нужно для работы с schoolMIPS

- Запуск симуляторе
  - компьютер с операционной системой
    Windows / Linux
  - ПО для симуляции
    Icarus Verilog / Modelsim
  - компилятор
    MIPS toolchain (gcc)



- Запуск на отладочной плате
  - ПО для синтеза и программирования ПЛИС
    Altera Quartus / Xilinx Vivado
  - отладочная плата ПЛИС
    от $50, поддерживается 10 отладочных плат
    перечень расширяется

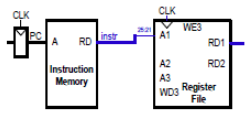


Все ПО доступно для свободной загрузки

Пример слайдов (пошаговое проектирование)

Процессор schoolMIPS: инструкция addiu

Шаг 2: считывание операндов-источников из регистрового файла

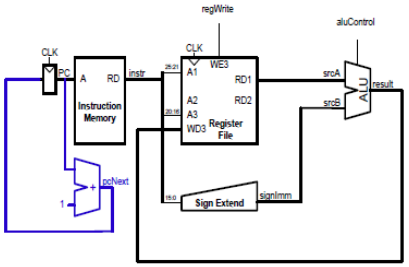


I-type. Integer Add Immediate, rt = rs + Immediate

31 op 26	25 rs 21	20 rt 16	15 Immediate 0

Процессор schoolMIPS: инструкция addiu

Шаг 6: вычисление адреса следующей инструкции

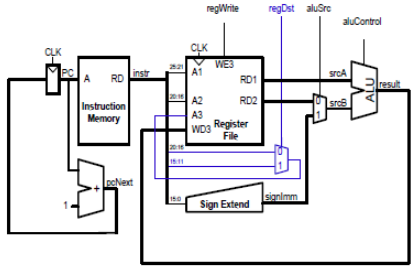


I-type. Integer Add Immediate, rt = rs + Immediate

31 op 26	25 rs 21	20 rt 16	15 Immediate 0

Процессор schoolMIPS: инструкция addu

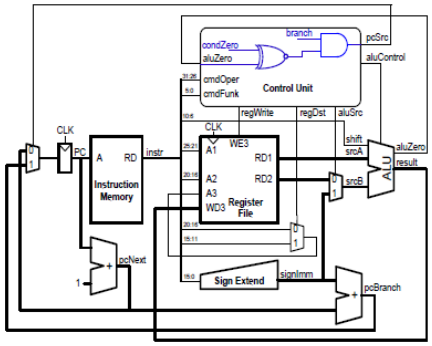
- определение регистра для записи результата
- запись результата вычислений



R-type. Integer Add Unsigned, rd = rs + rt

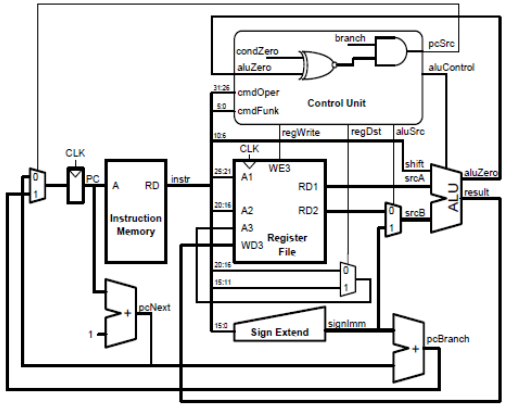
31 op 26	25 rs 21	20 rt 16	15 rd 11	10 sa 6	5 funct 0

Процессор schoolMIPS: инструкция beq



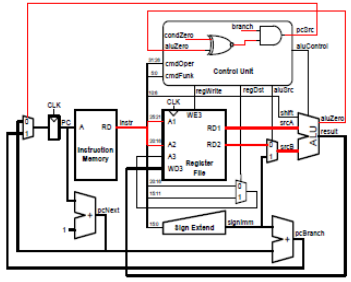
- определение необходимости перехода в зависимости от равенства результата нулю

Процессор schoolMIPS. Итоговая схема



Процессор schoolMIPS: сигналы управления (18)

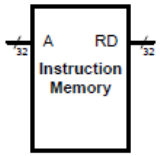
Instr	cmdOper	cmdFunc	branch	condZero	regDst	regWrite	aluSrc	aluControl
addiu	001001	??????	0	0	0	1	1	000
addu	000000	100001	0	0	1	1	0	000
srl	000000	000010	0	0	1	1	0	011
beq	000100	??????	1	1	0	0	0	101



Пример слайдов (реализация)

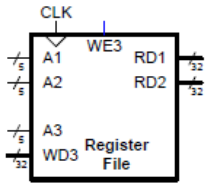
Реализация schoolMIPS. Память инструкций

```
// sm_cpu.v (line 35-37)
sm_rom reset_rom(pc, instr);
…
// sm_rom.v (line 2-17)
module sm_rom
#(
    parameter SIZE = 64
)
(
    input  [31:0] a,
    output [31:0] rd
);
    reg [31:0] rom [SIZE - 1:0];
    assign rd = rom [a];

    initial begin
        $readmemh ("program.hex", rom);
    end

endmodule
```



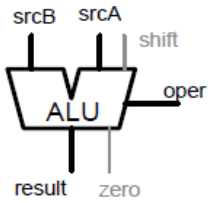
Реализация schoolMIPS. Регистровый файл

```
// sm_cpu.v (line 161-182)
module sm_register_file
(
    input  clk,
    input  [ 4:0] a0,
    input  [ 4:0] a1,
    input  [ 4:0] a2,
    input  [ 4:0] a3,
    output [31:0] rd0,
    output [31:0] rd1,
    output [31:0] rd2,
    input  [31:0] wd3,
    input  we3
);
    reg [31:0] rf [31:0];
    assign rd0 = (a0 != 0) ? rf [a0] : 32'b0; //for debug
    assign rd1 = (a1 != 0) ? rf [a1] : 32'b0;
    assign rd2 = (a2 != 0) ? rf [a2] : 32'b0;

    always @ (posedge clk)
        if(we3) rf [a3] <= wd3;
endmodule
```



Реализация schoolMIPS. ALU

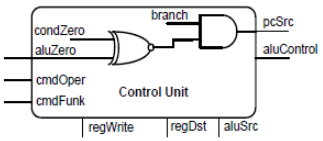
```
// sm_cpu.v (line 137-159)
module sm_alu (
    input      [31:0] srcA,
    input      [31:0] srcB,
    input      [ 2:0] oper,
    input      [ 4:0] shift,
    output            zero,
    output reg [31:0] result
);
    always @ (*) begin
        case (oper)
            default   : result = srcA + srcB;
            `ALU_ADD  : result = srcA + srcB;
            `ALU_OR   : result = srcA | srcB;
            `ALU_LUI  : result = (srcB << 16);
            `ALU_SRL  : result = srcB >> shift;
            `ALU_SLTU : result = (srcA < srcB) ? 1 : 0;
            `ALU_SUBU : result = srcA - srcB;
        endcase
    end
    assign zero = (result == 0);
endmodule
```



Реализация schoolMIPS. Устройство управления

Управляющие сигналы

```
// control unit (line 95-134)
module sm_control
...
    always @ (*) begin
        //control signals default values
        branch = 1'b0;
        condZero = 1'b0;
        regDst = 1'b0;
        regWrite = 1'b0;
        aluSrc = 1'b0;
        aluControl = `ALU_ADD;

        casez( {cmdOper,cmdFunk} )
          default                : ;
           { `C_SPEC, `F_ADDU } : begin
                                    regDst = 1'b1;
                                    regWrite = 1'b1;
                                    aluControl = `ALU_ADD;
                                  end

          ...
        endcase
    end
endmodule
```



Что делать со schoolMIPS

- Понять архитектуру
- Разобрать реализацию
- Запустить имеющиеся примеры

  счетчик, вычисление чисел Фибоначчи, квадратного корня

- Индивидуальный проект
  - добавление инструкции
  - расширение возможностей АЛУ
  - добавление памяти данных
  - подключение внешних устройств
  - многотактная реализация
  - конвейерная реализация
  - …

Опыт использования

- Курс «Системное проектирование цифровых устройств» МИЭМ, г.Москва

  преподаватель: Александр Романов

- Летняя Школа Юных Программистов 2017

  г.Новосибирск

  преподаватель: Юрий Панчул

- Школа-семинар в рамках конференции NGC 2017
  г.Томск

  преподаватель: Станислав Жельнио

Большое спасибо за внимание!
Ваши вопросы?