Университет ИТМО

Кафедра вычислительной техники

Отчёт по лабораторной работе № 4

по дисциплине: "Схемотехника ЭВМ"

Вариант №5

Студенты:
Куклина М.
Кириллова А.

Преподаватель: Баевских А.

Санкт-Петербург
2016 г.

Содержание

Цели работы

1. Знакомство с микроархитектурой конвейерного процессора MIPS32
2. Изучение принципов расширения системы команд конвейерного процессора

Описание структуры команд

CLZ

Format:

- [31 : 26] – spectial (011100)
- [25 : 21] – rs (source register)
- [20 : 16] – rt (the same as rd)
- [15 : 11] – rd (destination register)
- [10 : 6] – 00000
- [5 : 0] – 100001 (CLZ opcode)

Команда производит счёт количество лидирующих нулей в регистре rs и записывает результат в rd. Алгоритм:

```
temp = 32
for i in 32 .. 0 {
    if rs[i] == 1 {
        tmp = 31 - i
        break
    }
}
rd = temp
```

CLO

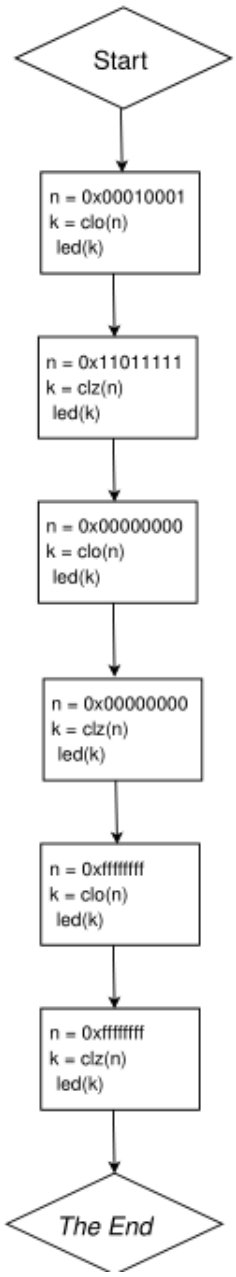
Format:

- [31 : 26] – spectial (011100)
- [25 : 21] – rs (source register)
- [20 : 16] – rt (the same as rd)
- [15 : 11] – rd (destination register)
- [10 : 6] – 00000
- [5 : 0] – 100000 (CLO opcode)

Команда производит счёт количество лидирующих единиц в регистре rs и записывает результат в rd.
Алгоритм:

```
temp = 32
for i in 32 .. 0 {
    if rs[i] == 0 {
        tmp = 31 - i
        break
    }
}
rd = temp
```

Блок-схема



Временные диаграммы

Сигналы:

1. result – результат операции;
2. b_in – входной операнд, над которым производится операция;
3. alu_ctl – код операции (clo – 11; clz – 10).

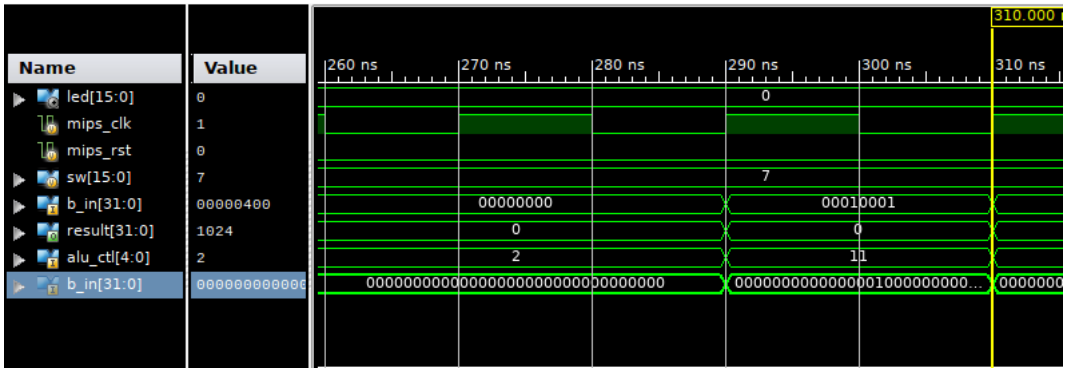


Рис. 1. Input: 0x00010001

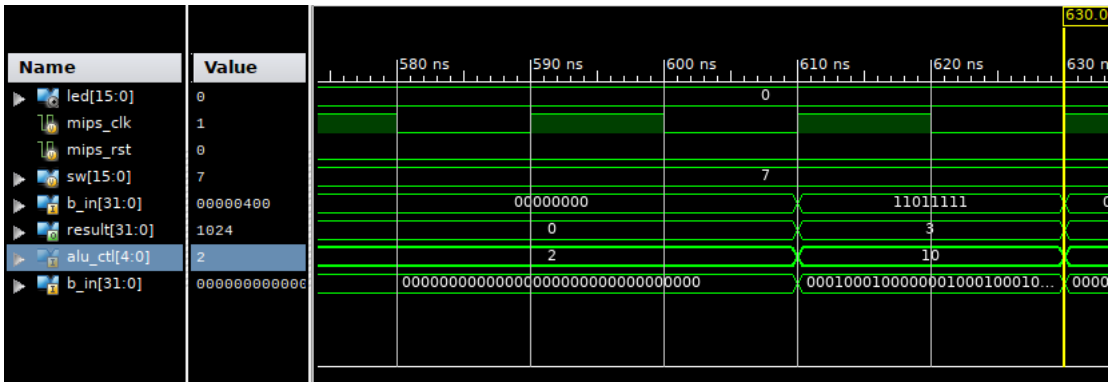


Рис. 2. Input: 0x11011111

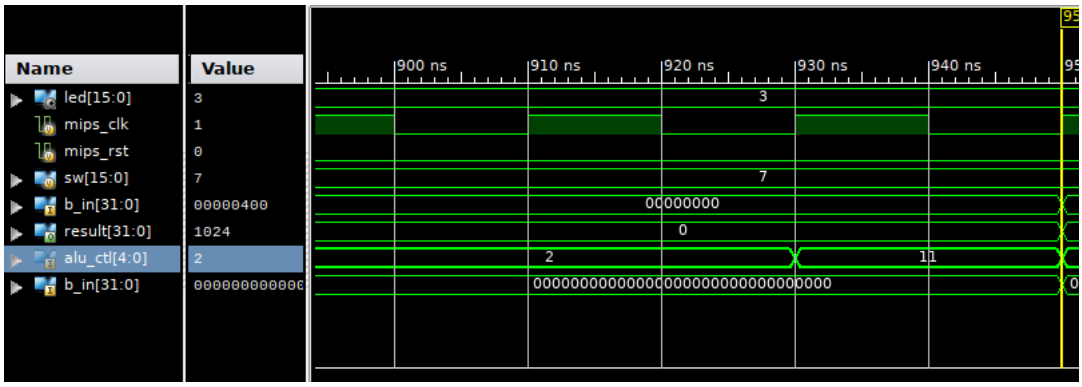


Рис. 3. Input: 0x00000000

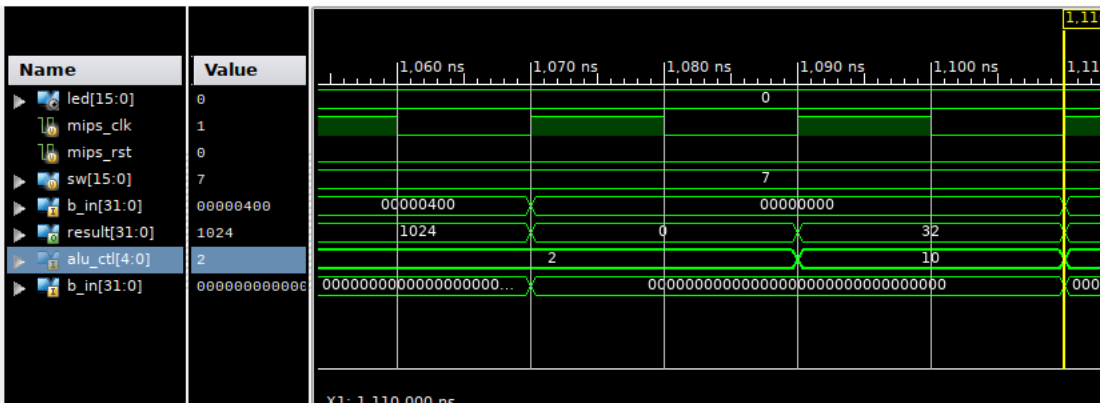


Рис. 4. Input: 0x00000000

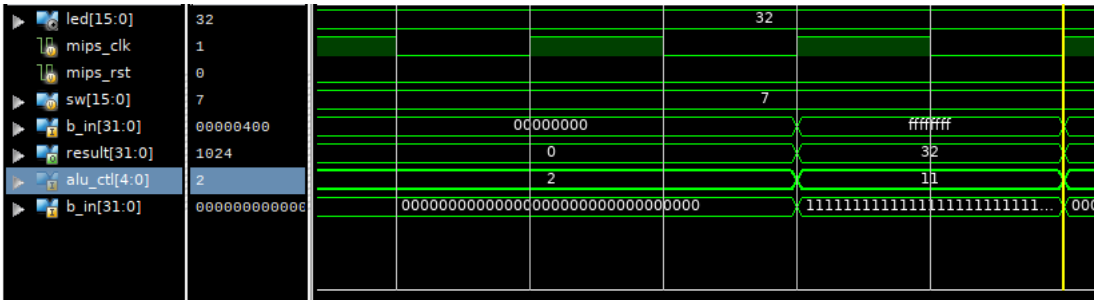


Рис. 5. Input: 0x11111111

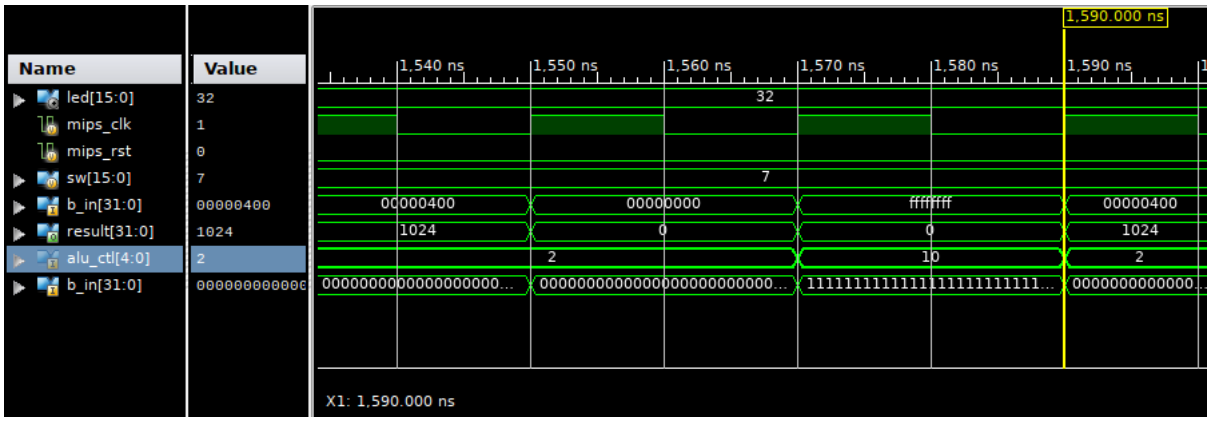


Рис. 6. Input: 0x11111111

Листинг

```
1  .global entry
2
3  .data
4          .word 0x00010001
5          .word 0x11011111
6          .word 0x00000000
7          .word 0xFFFFFFFF
8  .text
9  .ent entry
10 entry:
11     lw $t0, 0x200
12     clo $t1, $t0
13     sw $t1, 0x400
14     lw $t0, 0x201
15     clz $t2, $t0
16     sw $t2, 0x400
17     lw $t0, 0x202
18     clo $t1, $t0 */
19     sw $t1, 0x400
20     clz $t2, $t0
21     sw $t2, 0x400
22     lw $t0, 0x203
23     clo $t1, $t0
24     sw $t1, 0x400
25     clz $t2, $t1
26     sw $t2, 0x400
27 .end entry
```

В файле alu.v.

```
1 12,46c12
2 <
3 < /*
4 <       function integer clx;
5 <          input [31:0]b;
6 <          input var;
7 <          integer i;
8 <          begin
```

```
9 <             begin : CLXLOOP
10 <                 for (i = 0; i < 32; i = i + 1)
11 <                     if (var == b[31-i]) begin
12 <                         disable CLXLOOP;
13 <                     end
14 <             end
15 <             clx = i;
16 <         end
17 <     endfunction
18 < */
19 <     function integer clx;
20 <         input [31:0]b;
21 <         input var;
22 <         integer i;
23 <         reg a;
24 <         begin
25 <             a = 1;
26 <             clx = 0;
27 <             for (i = 0; i < 32; i = i + 1) begin
28 <                 if (a && var == b[31-i]) begin
29 <                     clx = clx + 1;
30 <                 end else begin
31 <                     a = 0;
32 <                 end
33 <             end
34 <         end
35 <     endfunction
36 <
37 ---
38 >
39 56,59d21
40 <                 10:                             // count leading zeros
41 <                     result = clx(b_in, 0);
42 <                 11:                              // count leading ones
43 <                     result = clx(b_in, 1);
44 63c25
45 <
46 ---
47 >
```

В файле control.v.

```
1 30d29
2 <                     SPEC   = 6'b011100;
3 38d36
4 <     reg register_op;
5 59d56
6 <         register_op   = ( opcode == SPEC );
7 99,104d95
8 <         else if (register_op) begin
9 <             ex_alu_op      = 2'b11; // operation defined by func code
10 <             ex_dst_reg_sel = 1'b1;  // rd dest
11 <             wb_mem_to_reg  = 1'b0;   // alu_out
12 <             wb_reg_write   = 1'b1;  // write to rd
13 <         end
```

Вывод

В ходе выполнения лабораторной работы были добавлены команды clo (count leading ones) и clz (count leading zeros). Обнаружилось, что предлагаемый к работе компилятор работает с подмножеством ассемблера MIPS, который не содержит в себе данные команды, так что при написании и комплияции кода приходилось переводить команды в машинный код вручную.