

Dec 06, 11 23:54

alu-main.c

Page 1/2

```

1  /*
2   alu-main.c
3   bho1 2006; bho1 19.11.2006; bho1 8.12.2007
4   bho1 29.11.2007 : init value for rega, regb, accu. flags to full 16 Bit
5   bho1 19.11.2009 : corrected error with neg_b calling neg_a
6   bho1 10.12.2009 : corrected error where neg_b and not_b loaded arg into rega
7   bho1 5.7.2011
8   renamed alu_exec_line to alu_parse_line
9   bho1 10.10.2011
10  functional approach: alu(ALU_OP_CODE, rega, regb, accumulator, flags);
11  GPL
12  */
13  #define _GNU_SOURCE
14
15  #include <stdio.h>
16  #include <stdlib.h>
17  #include <string.h>
18
19  #include "register.h"
20  #include "alu-opcodes.h"
21  #include "alu.h"
22
23  // the registers
24  char rega[REG_WIDTH+1] = "01234567";
25  char regb[REG_WIDTH+1] = "01234567";
26  char accumulator[REG_WIDTH+1] = "01234567";
27  char flags[REG_WIDTH+1] = "01234567";
28
29  /*
30   Reset ALU
31   resets registers and calls alu_op_reset
32  */
33  void alu_main_reset(char rega[], char regb[], char accumulator[], char flags[]){
34      int i;
35      /* clear rega, regb, accumulator, flags */
36      for(i=0; i<REG_WIDTH; i++){
37          rega[i] = '0';
38          regb[i] = '0';
39          accumulator[i] = '0';
40          flags[i] = '0';
41      }
42  }
43
44  /*
45   take string cmd_line, parse the line and call the alu function
46   corresponding to alu-opcodes.h
47  */
48  void alu_parse_line(char *cmd_line){
49      char opcode[100];
50      char operand1[100];
51      char operand2[100];
52      int nargs = 0;
53
54      nargs = sscanf (cmd_line, "%s%s%s", opcode, operand1, operand2);
55
56      switch(nargs){
57          case 3:
58              ldhex2register(operand1, rega);
59              ldhex2register(operand2, regb);
60              if(!strcmp(opcode,"add")){
61                  alu(ALU_OP_ADD, rega, regb, accumulator, flags);
62              }
63              if(!strcmp(opcode,"sub"))
64                  alu(ALU_OP_SUB, rega, regb, accumulator, flags);
65              if(!strcmp(opcode,"and"))
66                  alu(ALU_OP_AND, rega, regb, accumulator, flags);

```

Dec 06, 11 23:54

alu-main.c

Page 2/2

```

67         if(!strcmp(opcode,"or"))
68             alu(ALU_OP_OR, rega, regb, accumulator, flags);
69         if(!strcmp(opcode,"xor"))
70             alu(ALU_OP_XOR, rega, regb, accumulator, flags);
71         printf("%s%s%s\n", opcode, operand1, operand2);
72         break;
73     case 2:
74         ldhex2register(operand1, rega);
75         if(!strcmp(opcode,"neg_a"))
76             alu(ALU_OP_NEG_A, rega, regb, accumulator, flags);
77         if(!strcmp(opcode,"neg_b"))
78             alu(ALU_OP_NEG_B, rega, regb, accumulator, flags);
79         if(!strcmp(opcode,"not_a"))
80             alu(ALU_OP_NOT_A, rega, regb, accumulator, flags);
81         if(!strcmp(opcode,"not_b"))
82             alu(ALU_OP_NOT_B, rega, regb, accumulator, flags);
83         printf("%s%s\n", opcode, operand1);
84         break;
85     case 1:
86         if(!strcmp(opcode,"reset")){
87             alu_main_reset(rega, regb, accumulator, flags);
88             printf("%s%s\n", opcode, operand1);
89             break;
90         }
91         break;
92     }
93
94 }
95
96 void print_alu(char *rega, char *regb, char * accumulator, char flags[])
97 {
98     printf("Register A: ");
99     print_reg(rega);
100
101     printf("Register B: ");
102     print_reg(regb);
103
104     printf("Accumulator: ");
105     print_reg(accumulator);
106
107     printf("Carryflag: %c\n", getCarryflag(flags));
108     printf("Signflag: %c\n", getSignflag(flags));
109     printf("Zeroflag: %c\n", getZeroflag(flags));
110     printf("Overflowflag: %c\n", getOverflowflag(flags));
111     printf("*****\n");
112 }
113
114 int main()
115 {
116
117     size_t nbytes = 80;
118     char *cmd_line = (char *) malloc (nbytes + 1);
119
120     /*
121      read line from stdio, parse line, execute line, print result
122     */
123
124     while(getline(&cmd_line, &nbytes, stdin) != -1) {
125         alu_parse_line(cmd_line);
126         print_alu(rega, regb, accumulator, flags);
127     }
128
129     return 1 ;
130 }
131

```