Extended SSBC

ssbc.pl

```
#!/usr/bin/perl
###############################################################
#                           SSBC
# Simulator for SSBC.v4.r3.j3
# Docs: ssbc261-v4.r3.artn and testPlan
# File: ssbc.pl
# Author: Peter Walsh csci 261 Feb 2010
#          Updated Nov 25 2015 (PW)
#         Christian Paul Feb 28 2016
###############################################################

use Bit::Vector;
use Switch;

sub resetSSBC {

   open(MACHINECODE, "mac") || die "can't open mac file \n";

   $reset=0;

   my $vec = Bit::Vector->new(9);

   my $lc=0;
   while (<MACHINECODE>) {
      $instr=substr($_, 0, 8);
      $vec->from_Bin($instr);
      @mem[$lc] = $vec->to_Dec();
      $lc++;
   }

   close (MACHINECODE);

   $pc=0;
   $sp=0xfffa;
   $halt=0;
   $fault=0;
   #j additions
   $processed=0; #how many commands have been executed? (no no ops)
   $noops=0;     #how many no ops have been passed?
   $jumps=0;     #how many jumps
   $jumporigin=undef;
   $jumpdestination=undef;
}

sub printMenu {
```

```
   if ($debugmode) {
      printDebugMenu();
   } else {
      print "+------------------------+ \n";
      print "|  R: RESET              | \n";
      print "|  b: BREAK              | \n";
      print "|  r: RUN                | \n";
      print "|  A: READ PORT A        | \n";
      print "|  B: WRITE PORT B       | \n";
      print "|  C: READ PORT C        | \n";
      print "|  D: WRITE PORT D       | \n";
      print "|  s: STATUS             | \n";
      print "|  t: TOP                | \n";
      print "|  p: PSW                | \n";
      print "|  q: QUIT               | \n";
      print "|                        | \n";
      print "|  Enter menu selection: | \n";
      print "+------------------------+ \n";
   }
}

sub printDebugMenu {
   my $strpc = sprintf("%6s", $pc);
   my $strsp = sprintf("%6s", $sp);
   my $strsc = sprintf("%6s", 65530 - $sp);
   my $strjumps  = sprintf("%9s", $jumps);
   my $strjo  = sprintf("%6s", $jumporigin);
   my $strjd  = sprintf("%6s", $jumpdestination);
   print "+---------------+----------------------+---------------+\n";
   print "|  == CMDS ==   |  == CUSTOM CMDS ==   | == STATUS ==  |\n";
   print "|  R: RESET     |  S: BETTER STATUS    | PC:   $strpc  |\n";
   print "|  b: BREAK     |  P: SEE MEMORY       | SP:   $strsp  |\n";
   print "|  r: RUN       |  T: SEE STACK        | STACK:$strsc  |\n"; #
#CMDS: $processed
   print "|  A: R PORT A  |                      |               |\n"; #
LST CMD: $ir
   print "|  B: W PORT B  |                      |               |\n";
   print "|  C: R PORT C  |                      |  = JUMPS =    |\n";
   print "|  D: W PORT D  |                      | #: $strjumps  |\n";
   print "|  s: STATUS    |                      | SRC:  $strjo  |\n";
   print "|  t: TOP       |                      | DEST: $strjd  |\n";
   print "|  p: PSW       |                      |               |\n";
   print "|  q: QUIT      |  X: QUIT DEBUGGING   |               |\n";
   print "+---------------+----------------------+---------------+\n";
}

sub aluNor {
   my $ans, $xx, $yy;
   $xx=@_[0];
   $yy=@_[1];
```

```
   ($xx | $yy) ^ 0xff;

}


sub aluAdd {
   my $xx, $yy;
   $xx=@_[0];
   $yy=@_[1];

   ($xx+$yy) % 256;
}

sub aluSub {
   my $xx, $yy;
   $xx=@_[1];
   $yy=@_[0];

   if ($xx>=128) {
      $xx=($xx % 128);
      $xx=-$xx;
   }

   if ($yy>=128) {
      $yy=($yy % 128);
      $yy=-$yy;
   }


   if ($xx>=$yy) {
      ($xx-$yy) % 128 ;
   } else {
      ((($yy-$xx) + 128 ) % 256) ;
   }

}

sub adjustFlags {

   switch (@mem[0xfffb]) {
      case 0 {$Z=0; $N=0 }
      case 0x80 {$Z=1, $N=0 }
      case 0x40 {$Z=0, $N=1 }
   }
}

sub setFlags {
   my $xx;
   $xx=@_[0];

   @mem[0xfffb]=0;
```

```
   if ($xx==0) {
      $Z=1;
      @mem[0xfffb]=0x80;
   } else {
      $Z=0;
   }

   if ($xx>128) {
      $N=1;
      @mem[0xfffb]=0x40;
   } else {
      $N=0;
   }
}

sub insExe {

   switch ($ir) {
      case 0 { debugcmd("noop");
               $noops++;
               $processed--; } # no op
      case 1 { debugcmd("halt");
               $halt=1 }
      case 2 { debugcmd("pushimm " . @mem[$pc]);
               @mem[$sp]=@mem[$pc]; $sp--; $pc++ }
      case 3 { $ext=((@mem[$pc]*256)+(@mem[$pc+1]));
               debugcmd("pushext " . @mem[$ext] . " from " .
((@mem[$pc]*256)+(@mem[$pc+1])));
               @mem[$sp]=@mem[$ext]; $sp--; $pc+=2 }
      case 4 { debugcmd("popinh");
               $sp++ }
      case 5 { $ext=((@mem[$pc]*256)+(@mem[$pc+1]));
               debugcmd("popext " . @mem[$sp+1] . " to " .
((@mem[$pc]*256)+(@mem[$pc+1])));
               @mem[$ext]=@mem[$sp+1]; $sp++; $pc+=2;
                adjustFlags() if ($ext==0xfffb) }
      case 6 { debugcmd("jnz " . ((@mem[$pc]*256)+(@mem[$pc+1])));
               if (!$Z) {
                  $ext=((@mem[$pc]*256)+(@mem[$pc+1]));
                  $jumporigin=$pc; # debug mode
                  $pc=$ext;
                  $jumpdestination=$pc; # debug mode
                  $jumps++; # debug mode
                  debugcmd(">> jumped")
               } else { $pc+=2 } }
      case 10 { debugcmd("jnn " . ((@mem[$pc]*256)+(@mem[$pc+1])));
                if (!$N) {
                  $ext=((@mem[$pc]*256)+(@mem[$pc+1]));
                  $jumporigin=$pc; # debug mode
                  $pc=$ext;
```

```
                    $jumpdestination=$pc; # debug mode
                    $jumps++; # debug mode
                    debugcmd(">> jumped")
                  } else { $pc+=2 } }
      case 7 { debugcmd("add " . @mem[$sp+2] . " + " . @mem[$sp+1] . "
= " . aluAdd(@mem[$sp+2], @mem[$sp+1]));
                @mem[$sp+2]=aluAdd(@mem[$sp+2], @mem[$sp+1]); $sp++;
setFlags(@mem[$sp+1]) }
      case 8 { debugcmd("sub " . @mem[$sp+1] . " - " . @mem[$sp+2] . "
= " . aluSub(@mem[$sp+2], @mem[$sp+1]));
                @mem[$sp+2]=aluSub(@mem[$sp+2], @mem[$sp+1]); $sp++;
setFlags(@mem[$sp+1]) }
      case 9 { debugcmd("nor");
                @mem[$sp+2]=aluNor(@mem[$sp+2], @mem[$sp+1]); $sp++ }
   }
}

sub execSSBC {

   if ($pc >= $lc) {
      #print($lc);
      #printSurroundingMem();
   }

   if ((!$reset) && (!$fault)) {
      $ir=@mem[$pc];
      $pc++;
      if ($ir>$maxOpCode) {
         $fault=1;
         #print " +++ FAULT +++\n";
         #printSurroundingMem();
      } else {
         insExe();
         $processed++;
      }
   }
}

sub printmem {
   local $pointer = @_[0];
   local $comment = "";
   local $pointerVec = Bit::Vector->new(8);
   local $pointerStr = "";

   if (scalar(@_) > 1) {
      $comment = @_[1];
   }

   if (defined(@mem[$pointer])) {
      $pointerVec->from_Dec(@mem[$pointer]);
      $pointerStr=$pointerVec->to_Bin();
```

```
    }
    print "$pointerStr $comment\n";
}

sub printSurroundingMem {
    printmem($pc-4);
    printmem($pc-3);
    printmem($pc-2);
    printmem($pc-1);
    printmem($pc, "  <-- PC ($pc)");
    printmem($pc+1);
    printmem($pc+2);
    printmem($pc+3);
    printmem($pc+4);
}

sub printSurroundingStack {
    printmem($sp-4);
    printmem($sp-3);
    printmem($sp-2);
    printmem($sp-1);
    printmem($sp, "  <-- SP ($sp)");
    printmem($sp+1);
    printmem($sp+2);
    printmem($sp+3);
    printmem($sp+4);
    printmem($sp+5);
    printmem($sp+6);
}

sub debugcmd {
    if ($debug_outputcmds) {
        print sprintf("%-30s", @_[0]) . " (line: $pc)\n";
    }
}

# main

my $portaVec=Bit::Vector->new(8);
my $portcVec=Bit::Vector->new(8);
my $portbVec=Bit::Vector->new(9);
my $portdVec=Bit::Vector->new(9);
my $topStackVec=Bit::Vector->new(8);
my $pswVec=Bit::Vector->new(8);
my $sel;

$maxOpCode=10;
$reset=1;

#j additions
$debugmode = 0;
```

```
$debug_outputcmds = 0;

while (1) {
   printMenu();

   $sel=<>;
   chop($sel);
   switch($sel) {
      case "R" { resetSSBC() }
      case "b" { if ((!$halt) && (!$fault)) { execSSBC() } }
      case "r" { if ($reset == 1) { resetSSBC() }
                 while ((!$halt) && (!$fault)) { execSSBC() } }
      case "B" { print "Enter Port B value in binary (8 bits) ";
                 $portbStr=<>;
                 $portbStr=substr($portbStr, 0, 8);
                 $portbVec->from_Bin($portbStr);
                 @mem[0xfffd]=$portbVec->to_Dec() }
      case "A" { if (defined(@mem[0xfffc])) {
                    $portaVec->from_Dec(@mem[0xfffc]);
                    $portaStr=$portaVec->to_Bin();
                 }
                 print "Port A value: $portaStr \n" }
      case "D" { print "Enter Port D value in binary (8 bits) ";
                 $portdStr=<>;
                 $portdStr=substr($portdStr, 0, 8);
                 $portdVec->from_Bin($portdStr);
                 @mem[0xffff]=$portdVec->to_Dec() }
      case "C" { if (defined(@mem[0xfffe])) {
                    $portcVec->from_Dec(@mem[0xfffe]);
                    $portcStr=$portcVec->to_Bin();
                 }
                 print "Port C value: $portcStr \n" }
      case "s" { print "Fault: $fault \n Halt: $halt \n" }
      case "t" { if (defined(@mem[$sp+1])) {
                    $topStackVec->from_Dec(@mem[$sp+1]);
                    $topStackStr=$topStackVec->to_Bin();
                 }
                 print "Top of Stack: $topStackStr \n" }
      case "p" { if (defined(@mem[0xfffb])) {
                    $pswVec->from_Dec(@mem[0xfffb]);
                    $pswStr=$pswVec->to_Bin();
                 }
                 print "PSW: $pswStr \n" }
      case "q" { exit() }

      case "P" { printSurroundingMem() }
      case "S" { print "Fault: $fault \n Halt: $halt \n";
                 print "   PC: $pc \n   SP: $sp \n";
                 print "Last Command: $ir \n# of processed cmds:
$processed \n";
                 print "# of no ops: $noops \n" }
```

```
        case "T" { printSurroundingStack() }
        case "X" { if ($debugmode) {
                       $debugmode = 0;
                       $debug_outputcmds = 0;
                   } else {
                       $debugmode = 1;
                       $debug_outputcmds = 1;
                   } }
    }
}
```

From:
https://wiki.chrpaul.de/ - **ChrisWiki**

Permanent link:
https://wiki.chrpaul.de/projects:ssbc

Last update: **2016/03/03 21:51**