Assignment 3 : MIPS implementation by VHDL

We created 2 arrays of std_logic_vectors(31 downto 0),
1. mem1 with size 4096
2. reg1 with size 32

reg1 array is used to store the values of all 32 registers according to their corresponding indexes.

Mem1 array is memory. We store MIPS instructions (.text) in 0 to 1024 index of mem1 array and the rest part of mem1 (1025 to 4095) is used for .data part.

We defined 7 states which change in each other after each clock event.

Initially system is in idle state,

here we take one bit_vector (instruction) from mem1 array and change state to read.

Read state :

first we confirm whether the command is of R type or I type. If R type, then we load rsrc1, rsrc2, rdest, shamt, funct singals in this way:

```
rsrc1<=to_integer(unsigned(temp(25 downto 21)));
   rsrc2<=to_integer(unsigned(temp(20 downto 16)));
   rdest<=to_integer(unsigned(temp(15 downto 11)));
   shamt <= to_integer(unsigned(temp(10 downto 6)));
   funct <= to_integer(unsigned(temp(5 downto 0)));
```

here temp is the bit_vector of that instruction.

if shamt is not zero, then it must be a shift operation, so we jump to shiftoper state. we assign rtype to 1 and change our state to change (if R type)
if the instruction is I type,
then we load ifirst, irs, ird, iaddress1 states in this way:

```
ifirst<= to_integer(unsigned(temp(31 downto 26)));
irs <= to_integer(unsigned(temp(25 downto 21)));
ird <= to_integer(unsigned(temp(20 downto 16)));
iaddress1 <= to_integer(unsigned(temp(15 downto 0)))/4;
```

And state jump to updateir.

Shiftoper state :

now if funct is 0 then it is shift left operation, so we jump to shiftlefthelper state.
Otherwise if funct is 2 then it is shift right operation, so we jump to shiftrighthelper state.
Else we jump to idle state.

Shiftlefthelper state :

in this state, by each clock event, we shift the register value to left by 1 bit, and decrease shamt by 1, until shamt reaches to 0.
after this, state jump to idle.

Shiftrighthelper state :

in this state, by each clock event, we shift the register value to right by 1 bit, and decrease shamt by 1, until shamt reaches to 0.
after this, state jump to idle.

Updateir state :

in this we gave value to irs1 by to_integer(unsigned(reg1(irs))) + iaddress1 and change the state to 'change';

change state :

here we first check the type of intruction.

R type :

if temp(5 downto 0)="100000" then it is an add instruction.
So we assign the sum of reg1(rsrc1) and reg1(srsc1) to reg1(rdest) and change our state to idle.
If temp(5 downto 0)="100010" then it is an substract instruction.
So we assign the substraction of reg1(rsrc1) and reg1(srsc1) to reg1(rdest) and change our state to idle.

I type :

if ifirst is 35, then it is a lw command, so we assign the value of mem1(irs1) to reg1(ird).
If ifirst is 43 then it is a sw command, so we assign the value of reg1(ird) to mem1(irs1).

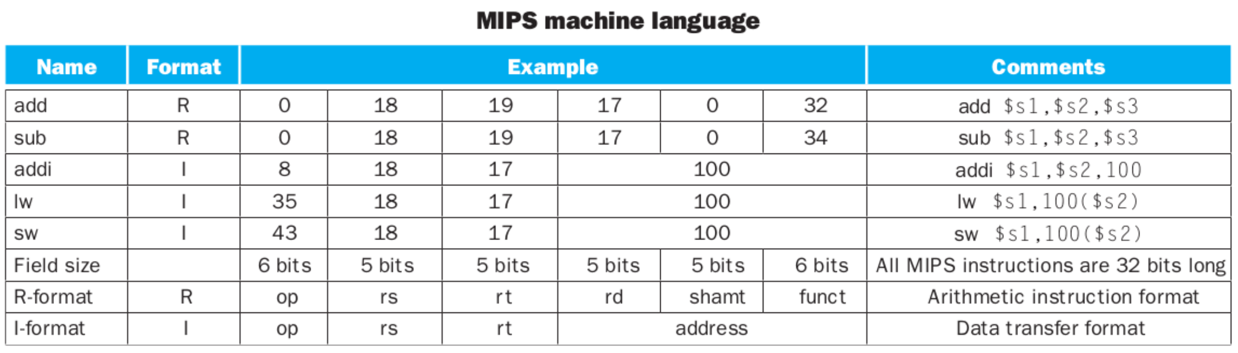
Finally the state changes to idle.

Binary input for MIPS instructions:
for shift left by 4 bit:

op	rs	rt	rd	shamt	funct
0	0	16	10	4	0

for shift right by 4 bit:

op	rs	rt	rd	shamt	funct
0	0	16	10	4	2



MIPS machine language

Name	Format	Example						Comments
add	R	0	18	19	17	0	32	add `$s1,$s2,$s3`
sub	R	0	18	19	17	0	34	sub `$s1,$s2,$s3`
addi	I	8	18	17	100			addi `$s1,$s2,100`
lw	I	35	18	17	100			lw `$s1,100($s2)`
sw	I	43	18	17	100			sw `$s1,100($s2)`
Field size		6 bits	5 bits	5 bits	5 bits	5 bits	6 bits	All MIPS instructions are 32 bits long
R-format	R	op	rs	rt	rd	shamt	funct	Arithmetic instruction format
I-format	I	op	rs	rt	address			Data transfer format

Manoj Kumar 2018CS50411
Nikita Bhamu 2018CS50413