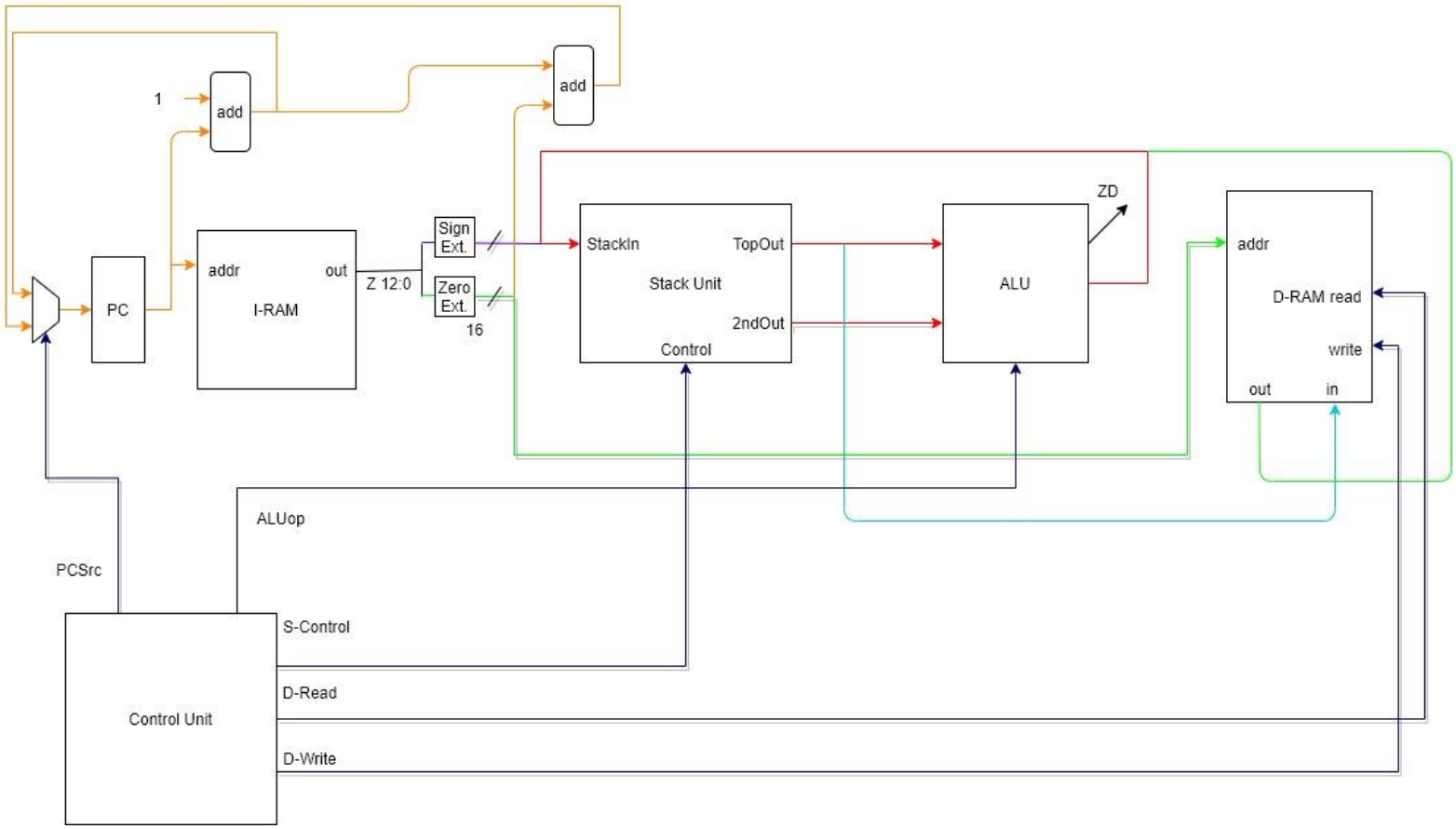

	PCSrc	ALUop	S-Control	D-Read	D-Write
add	0	‘add’	100	0	0
nand	0	‘nand’	100	0	0
push	0	0	010	1	0
pop	0	0	001	0	1
pushi	0	0	010	0	0
beq	0 / 1	‘sub’	000 / 011	0	0
noop	0	0	000	0	0
halt	0	0	000	0	0

vSaw2 datapath design & control table

Designer: Haoyuan Du

Description:

In the vSaw2 datapath illustration, there are one control unit, one PC unit, three ALU, one MUX, one sign-extended, one zero-extended, one Instruction RAM, one Data RAM and one Stack Unit.

There are 8 instructions implemented. Firstly, ‘and’ & ‘nand’ instruction are similar. Both operands are on the Stack Unit located in ‘TopOut’ and ‘2ndOut’ respectively. When S-Control gives ‘100’(binary) signal (pop top 2 values and push the value ‘stackIn’ onto the top of the stack), both values will be popped and sent to ALU that was given an ‘add’ or ‘nand’ operation respectively. The answer from ALU will be sent to ‘StackIn’ port and push onto the Stack Unit. The red lines in the graph illustrate both instructions. Lastly, increase PC by 1 by sending PCSrc signal ‘0’. The MUX will choose the answer of ‘PC+1’ and update it to the PC unit (orange lines).

Next, the ‘push’ instruction. At the beginning, 13 bits of Z is sent out from I-RAM. Because it’s an address, we need to use ‘zero-extended’ to get a 16 bits address. Then, the address is sent to D-RAM so that D-RAM knows the current address. Simultaneously, D-Read is on(1) so that the value on current address is sent from the ‘out’ port to the ‘StackIn’ port (Stack Unit) through green lines. By the ‘010’(binary) signal at S-Control, we push the value in ‘StackIn’ port onto the top of the stack. Lastly, increase PC by 1 by sending PCSrc signal ‘0’. The MUX will choose the answer of ‘PC+1’ and update it to the PC unit (orange lines).

Thirdly, the ‘pop’ instruction. Similarly as above, giving S-Control signal as

‘001’(binary), the ‘TopOut’ value is sent to D-RAM’s ‘in’ port through blue lines. The 13 bits of Z (address) is zero-extended and sent to D-RAM’s ‘addr’ port. By sending D-Write signal as 1, the value will be placed into the memory location specified by 16 bits of Z. Lastly, increase PC by 1 by sending PCSrc signal ‘0’. The MUX will choose the answer of ‘PC+1’ and update it to the PC unit (orange lines).

Fourthly, the ‘pushi’ instruction. We get Z value from I-RAM as usual, using ‘sign-extended’ to get a 16 bits version of Z because it's an immediate value, not an address anymore. Then, sending it through violet lines to the ‘StackIn’ port (Stack Unit). S-Control signal is ‘010’(binary), so we push the immediate Z value onto the stack. Lastly, increase PC by 1 by sending PCSrc signal ‘0’. The MUX will choose the answer of ‘PC+1’ and update it to the PC unit (orange lines).

Fifthly, the ‘beq’ instruction. The first step, we send both ‘TopOut’ and ‘2ndOut’ values to the ALU and let ALU do the subtraction. If the ZD (zero detected) doesn’t detect zero (ZD=0), then we give ‘000’(binary) to the S-Control and give ‘0’ to the PCSrc. In this case, we do nothing on the stack except increasing PC by 1. On the other hand, if ZD=1, then we give ‘011’(binary) to the S-Control and give ‘1’ to the PCSrc. That means we pop both values from stack and branch to the location of PC+1+Z through orange lines.

For the ‘noop’ instruction, we only increase PC by 1 (orange lines) by sending ‘0’ to the PCSrc. Other control signals are regardless so we convert them to 0 according to the pdf.

Finally, the ‘halt’ instruction, we also increase PC by 1 (orange lines) by sending

‘0’ to the PCSrc. Then, halt the machine by sending all zeros from the Control Unit.

In orange lines area, both ALU only need one operation ‘add’ which can save more space and money.

In addition, repeated electronic lines are omitted by sharing one electronic line. For example, the blue line is a branch of the red line which comes from the ‘TopOut’ port. They have the same value at that time, but can go different destinations. To do this, we save money as well.