ECE369A Computer Organization
LAB 7

- **Objective:**
  - Execute your vbsme on the pipelined datapath with hazard detection
- **Eligibility**
  - At least 75% of the instructions are functional on the datapath (Labs4-5)
  - At least 75% of the private test cases for pipelining with hazard detection passing (Lab 6)
  - Equal contribution on Lab 3 for full score. If unequal, Lab7 score will be scaled proportionally.
  - Equal contribution on Labs4-5 for full score. If unequal, Lab7 score will be scaled proportionally.
- **Method**
  - Set DivVal in your clock divider to 10,000 to speed up the display rate to get through the test cases quickly, while still being slow enough to read the answers. You may want to start with single test case and then scale to 5 test cases.
  - Use the provided vbsme.s template file that includes 5 test cases and include your vbsme function.
  - Generate Instruction_memory.mem and Data_memory.mem using your vbsme.s with the help of MipsHelper
  - There are multiple changes needed to the vbsme.s before passing through the MIPSHelper
    - vbsme template includes instructions such as “la” and “li”. You need to replace them with “addi” or any other suitable instruction supported by your datapath.
    - You need to comment out the “jal print_result” instructions
    - You need to comment out the “print_result” subroutine
    - Include an infinite loop after completing the 5 test cases: “end_program: j end_program”
  - Submit the code with Instruction_memory.mem and Data_memory.mem. Your Instruction_memory.v and Data_memory.v modules should initialize memories using $readmemh and the two .mem files.
  - During Demo, TA will download your project from D2L that includes all .v and .mem files along with constraint file, synthesize your design and execute the program on the **FPGA**
    - The (X,Y) coordinates of the block of the current minimum SAD should be displayed on the FPGA.
    - Display will start with (0,0) and each time a block with smaller SAD is found new coordinates should be displayed
  - **Offline testing using private cases to be conducted by TAs**
- **Deliverable**: Following files are required as part of your submission:
    - All design and testbench Verilog files: "*.v" along with constraint files
    - Data file used for initializing data and instruction memory (*.mem)
    - .s (final form of your vbsme)
- **Penalty Conditions:**
  - Percent effort not reported (15% penalty)
  - Late submission (15% per day)
  - Submitting files in a folder or in compressed form (zip/tar). (15% penalty)
  - Changing the file name or extension. (15% penalty)
  - Missing any required such as .mem, .s , .v file (refer to “Deliverable” item above) 50% penalty
  - Failing to demonstrate (80% penalty)
  - Design works in behavioral simulation but fails to synthesize (80% penalty)

    - Design works in behavioral simulation, synthesizes with warnings but post-implementation functional simulation fails (70% penalty)
    - Design works in post-implementation functional simulation, but FPGA fails to display (30% penalty)
    - All team members must attend the demonstration
        - Unable to answer questions about your implementation during demo – up to 75% penalty
        - Not attending the demo (90% penalty)

For clarification questions from past years that might be applicable refer to next page. Please note that some of the procedures may change to improve the flow of the demonstrations. Please check D2L for updates

Questions From Previous Years

Q1. If you need to use stack for your SAD routing:
Stack will be implemented in data memory. On a Reset, you can initialize $sp to the last location in the data memory.

Q2. JAL
JAL implementation and register update
For project, while implementing JAL do not add additional write port to MIPS register file (32 register file). JAL should update register $ra in the write-back stage only.

Q3. Test Cases
Sharing testcases
I was wondering if it would be alright if each group wrote their own test cases and stick them in a huge repository (git or google docs). It would help out in trying to cover the multitude of cases that can be present for this part of the lab.

We highly encourage sharing test cases.

Q4. reading instruction_memory.mem
I placed my .txt file for instruction memory in the simulation sources folder as suggested in another post, but I still can't seem to get vivado to find it. Anyone else having this issue?

You should put this in to design sources folder. Or try adding it into project in the vivado. You will have to follow similar steps that you use for adding RTL file.

In Vivado:
Add Sources --> Add or Create Simulation Sources ---> Change file type to "All Files" --> Select "Instruction_memory.mem"

Also, you may have to right click on the text file and change the type of file it is to "Data" before it can be read by your instruction memory. I had that issue even after I loaded the file into the project.

Another to ensure Vivado is using the correct file is to use the absolute path when calling $readmemh (ex. $readmemh("C:\Users\akoglu\ece369\lab7\instruction_memory.mem")). When Vivado sees multiple memory files with the same name, it's not always clear which one it will use.

Q5. MIPS Helper
Creating Instruction memory to be read by readmemh
I can't seem to get the mipshelper to output hex instructions without comments. I really don;t want to go through and delete each line, so has anyone figured out how to get it to output just hex similar to "Instruction_memory.mem" that was given to us on d2l?

You can use following commnad:
1) Run MipsHelper.

```
./mipsHelper369 -aiohd input.s  output.txt
```

2) On ECE node open the text file in "VI" editor by typing "vi output.txt" on the shell then hit enter. It will open the file in VI editor.

in VI editor just type this ":%s/ ->.*$//g" and hit enter. This will do the required changes. Save file by typing ":wq!" and then hit enter.

FQ6. FPGA Board
Output to the Board

What exactly will be the required output to the FPGA board per cycle?
For SAD on FPGA, it should be the registers tracking the coordinates (X,y) for the current minimum SAD location during the execution.

Q7. What about when we reach the end of the program? Should we put an infinite loop with a nop inside at the end of our program or something, so that we don't keep fetching more instructions?

Implement an indefinite loop like the following:
here: j here

Q8. Clock Divider
Is a Clock divider module necessary? I saw in the provided xdc file the following line:
create_clock -period 100.000 -name Clk -waveform {0.000 5.000} [get_ports Clk]
would changing the values in this line in the xdc essentially function to adjust the clock speed to the required setting? If so, which values would I have to change, and which values do what?

xdc file is not the place to create the clock divider. The xdc file is used for applying design constraint. Above clock command is used by the compiler as follows:

create_clock : it indicates the signal named as "Clk" in top module is a clock signal.

-period: With this input, compiler will try to synthesize and route design on FPGA with maximum critical path of 100ns. It indicates the maximum operating frequency for your design (1/period)Hz. For competition you will have to play with this parameter to determine the minimum critical path acceptable for your design. Making it too small will give you incorrect post-implementation behavior.

-waveform: It indicates the "Clk" duty cycle. In above case it indicates that if clock starts at 0 ns then positive edge will occur at 0ns and next negedge on clock will occur on 5ns. You can change 5ns to 50ns to indicate 50% duty cycle but it wont affect your critical path. critical path will be dependent on "-period".

You will have to write a RTL module to create a clock divider which will use above clock as a reference clock input.

Q9. Instruction Memory
instruction_memory.mem wrong on “la” instruction
In the instruction_memory.mem - la $s2, asize1: We know that the la instruction is the equivalent of lui. However, if you convert the hex instruction to binary and compare the opcode to what lui is supposed to be, it is different. The opcode turns out to be the opcode for an ori instruction, which causes unexpected results. Is this an error with the MIPSHelper?

In qtspim, we can not access data memory at address location 0 as initial address space is reserved for system specific code so we need to use "la". In your HDL implementation, memory location 0 is accessible to you. So mipsHelper will take the starting memory address as location 0 and will convert "la" to "ori" to load correct memory address.

Q10 could not open $readmem data file "Instruction_memory.mem"
[Synth 8-4445] could not open $readmem data file 'Instruction_memory.mem'; please make sure the file is added to project and has read permission, ignoring ["C:/Users/netid/Downloads/InstrMem.v":923]

I'm getting this error after implementation, although the Instruction_memory.mem is in simulation sources and is a memory initilization file. Does anyone know the fix?

It has to be added as a design source as well for it to work for synthesizing and implementation.

EDIT: you also might have to right click on the memory files after they're added to the project and set the type as memory initialization files.

Q11. Data Memory
Phase 1 Data Memory
The project requirements state we only need to read from the .mem file for the instruction memory. What should data memory be initialized to for the set of private test cases we are getting for the phase 1 test? Do we leave the data memory as it is from our own tests, Set it to all zeroes, or does it also need to read from some data_memory.mem etc. ?

You will be reading data memory from a memory file. Name it as "data_memory.mem".

When creating your own testcases .s file, the data memory may come from the lines at the top of the file before '.text'. You can use MipsHelper to convert those lines into a datamemory.mem file by running:
./mipsHelper -aiodh ./testcases.s ./out.txt
This will produce two files, 'out.txt' and 'out_data.txt'. Since the instruction memory is already given to us in the proper format, you can ignore the first. The second will essentially be a Verilog file that you could copy into the data memory module. However, in order to make it readable, (it should be fairly short anyways) you should take each line and replace 'memory[n] = 32'h100' (for example) with the plain hexadecimal number, 00000064.

Q12. initialize Data memory
Does anyone know what value to initialize the data memory in order to get the list to load?

```
 .data
asize0:  .word    100, 200, 300, 400, 500, 600
asize1:  .word    700, 800, 900, 1000, 1100, 1200
.text
```

If you are looking to create the file for data Memory you can use the mipsHelper with the '-d' command and it will create a data memory file in addition to the instruction memory file, for the demo you will need to read from a file so I recommend that way. If you want to hard code a few values the memory of asize0 starts with position 0, and increments up from there. So it would look like: memory[0] = 100; memory[1] = 200; etc and asize1 starts at position 7.

Q13. Timing Analysis
I am currently trying to run the timing analysis of our datapath and I am following the outline provided in the Verilog folder in D2L and it is showing a Clock in step 8). Is that clock from the Clock Divider in our project? I don't seem to have that in my timing analysis which makes me believe that I currently don't have an external clock running my project.

You can put same timing constraints on both the clock, input clock to your wrapper and output of clock divider (input to your datapath). But only input clock to wrapper should be assigned a pin number on the FPGA.

Q14. Issue with $readmemh and Instruction Memory
We're trying to read Instruction_memory.mem into our InstructionMemory module, but we keep running into this error, no matter how we add the text file to our project.

"could not open $readmem data file 'Instruction_memory.mem"; please make sure the file is added to project and has read permission"

We've already added instruction_memory.mem to the simulation sources, but the same error still shows. Has anyone else run into this problem?

download file> add sources>add design source>(find the file)> all files(dropdown)> finish>(at this point the file should be in a folder in your project called text files)> open that folder> right click file> setfile type> memory initialization file

Q15. Question on $readmemh
The notation given to us was "$readmemh ("Instruction_memory.mem", <memory register identifiers>)" and some lines of code in the text file are:

```
3c120000
8e520000
3c130000
8e730004
02538820
```

Which are hexadecimal values. Does that mean $readmemh ONLY accepts hex values? I've been trying to put my own binary values such as:

```
00100000000100000000000000000001
00000000000000000000000000000000
00000000000000000000000000000000
00000000000000000000000000000000
00000000000000000000000000000000
00000000000000000000000000000000
00100000000100010000000000000001
```

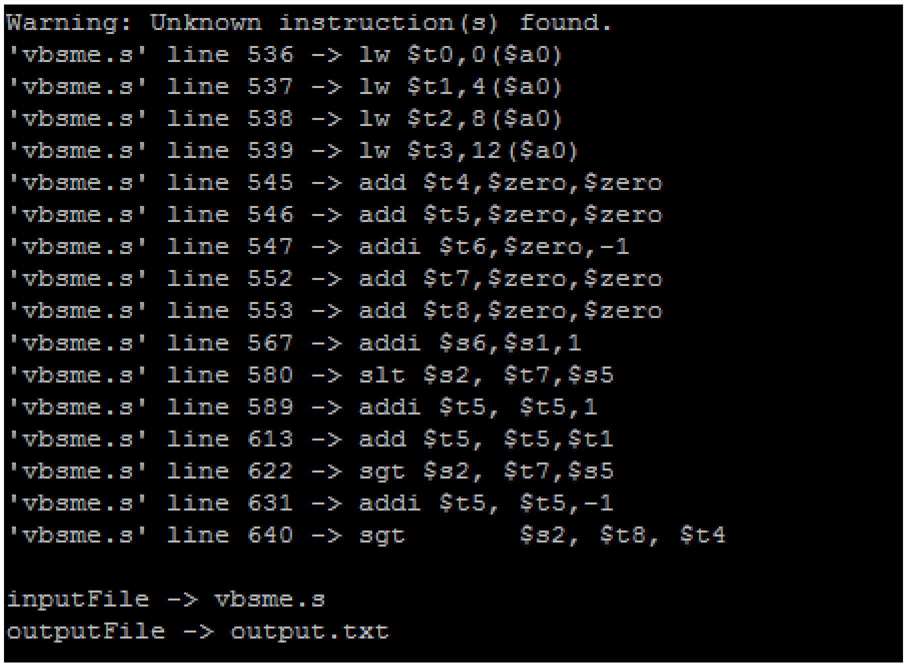
to be read but it isn't processing correctly. Do I simply need to convert these binary values to hex values for it to be read by $readmemh correctly? I would think $readmemh would be able to read the value in the text value no matter what notation it's in.

readmemh: to read hexadecimal
Also, you can use the h flag in mipshelper to have the output be in hex.

Q16. test case wrong?
the ori messes up my execution. I've come to the conclusion that I've either gone crazy, or something is wrong in the file. The ORI instruction uses 0xaaaa as the immediate field, and I searched for this value in hex, and decimal, and couldn't find it in my entire datapath... it doesn't exist. The instruction is correct in my memory. Anyone else run into this issue?

Andi ori and xori all are zero extended and not sign extended.

Q17. Mips helper giving errors on LW
When trying to convert our VBSME I am getting unknown instructions on known instructions. looking for advice.



```
Warning: Unknown instruction(s) found.
'vbsme.s' line 536 -> lw $t0,0($a0)
'vbsme.s' line 537 -> lw $t1,4($a0)
'vbsme.s' line 538 -> lw $t2,8($a0)
'vbsme.s' line 539 -> lw $t3,12($a0)
'vbsme.s' line 545 -> add $t4,$zero,$zero
'vbsme.s' line 546 -> add $t5,$zero,$zero
'vbsme.s' line 547 -> addi $t6,$zero,-1
'vbsme.s' line 552 -> add $t7,$zero,$zero
'vbsme.s' line 553 -> add $t8,$zero,$zero
'vbsme.s' line 567 -> addi $s6,$s1,1
'vbsme.s' line 580 -> slt $s2, $t7,$s5
'vbsme.s' line 589 -> addi $t5, $t5,1
'vbsme.s' line 613 -> add $t5, $t5,$t1
'vbsme.s' line 622 -> sgt $s2, $t7,$s5
'vbsme.s' line 631 -> addi $t5, $t5,-1
'vbsme.s' line 640 -> sgt       $s2, $t8, $t4

inputFile -> vbsme.s
outputFile -> output.txt
```

you need to insert space in between registers for each instruction.
for example instead of : lw $t0,0($a0) it should be: lw $t0, 0($a0). pay attention to space between $t0 and 0($a0),