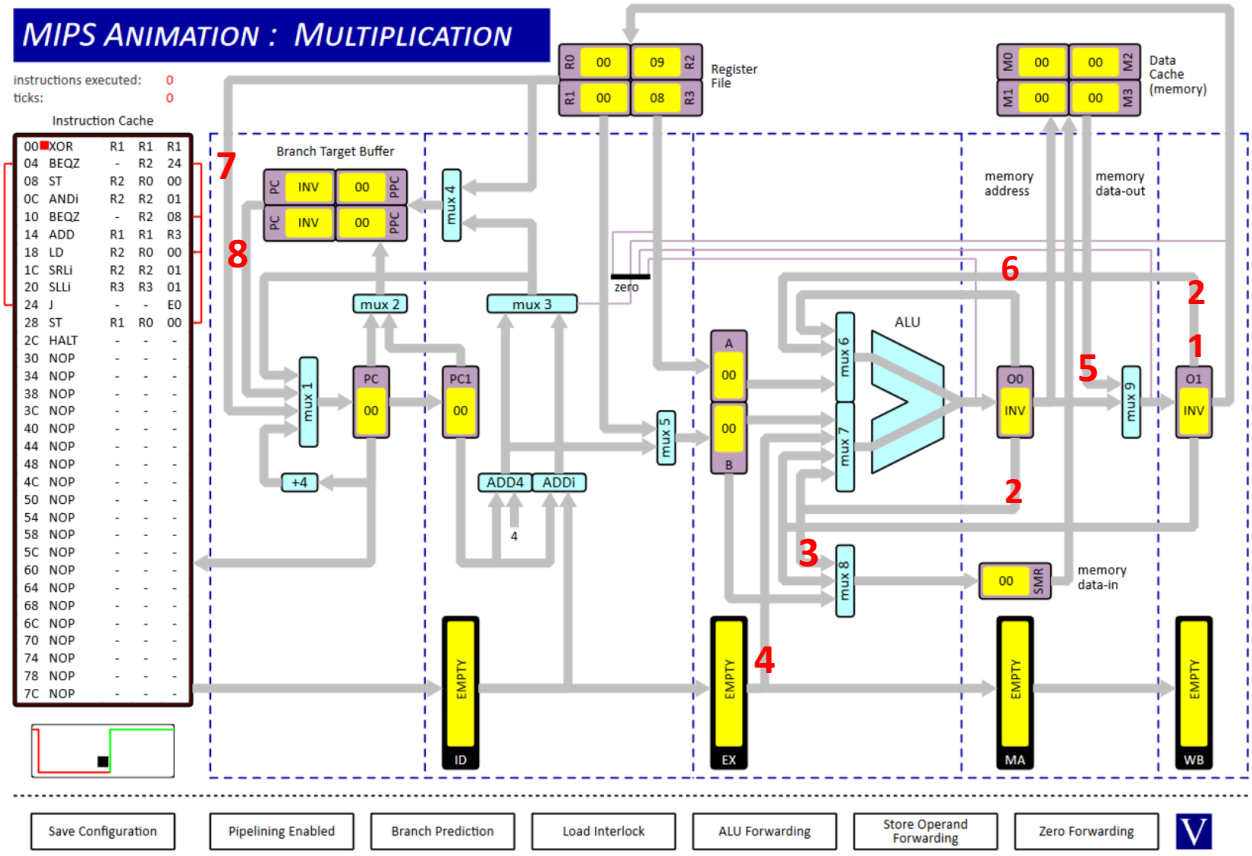
CSU34021 Tutorial 4

You will need to use the DLX/MIPS animation to help you answer these questions (http://www.scss.tcd.ie/Jeremy.Jones/VivioJS/vivio.htm).

Q1. The figure below shows the internal data paths of the DLX/MIPS processor.



For each sub-question below, give a short code segment (one or two instructions) which shows the specified data paths(s) being used. Paste a screen shot into your answer.

1. O1 to MUX6
2. O0 to MUX7 and O1 to MUX6 (simultaneously)
3. O0 to MUX8
4. EX to MUX7
5. Data cache to MUX9 (memory data-out)
6. O0 to Zero detector
7. Register File to MUX1
8. Branch Target Buffer to MUX1

Note that you can store your program in the DLX/MIPS program database by clicking on "save configuration". Save under "CSU34021/your username/q1" e.g. "CSU34021/jones/q1".

Q2. Consider the execution of the following code segment (initially r1 = 1 and r2 = 2):

```
add     r1, r1, r2      ; r1 = r1 + r2
add     r2, r1, r2      ; r2 = r1 + r2
add     r1, r1, r2      ; r1 = r1 + r2
add     r2, r1, r2      ; r2 = r1 + r2
add     r1, r1, r2      ; r1 = r1 + r2
halt
```

Determine the resulting value of r1 and the number of clock cycles needed to execute the code segment if (i) ALU Forwarding is enabled (ii) ALU forwarding is disabled with CPU data dependency interlocks enabled and (iii) ALU forwarding with CPU data dependency interlocks disabled. Explain in detail why the results and number of clock cycles are different.

You can click "Instruction Cache" until the program above is displayed.

Q3. Click "Instruction Cache" until the program shown in Q1 is displayed. The program calculates r1 = r2*r3 (0x48 = 0x09 x 0x08)

(i) How many instructions are executed and how many clock cycles are needed to execute this program until it halts? Explain in detail why these two numbers are not equal and account for each stall cycle.

(ii) Click "Branch Prediction" until "Branch Interlock" is displayed. How many cycles are now needed to execute this program until it halts? Explain in detail why this number differs from your answer to part (i).

(iii) Using “Branch Prediction”, what is the effect on execution time if the two shift instructions are swapped and why?