**Project 5 – Pipelined MIPS**

Programmer: Victor Espinoza

SID: 010657450

CECS 440, Section 5; Tu/Th 9:30 - 11:45 A.M.

Lab Section 6

Due: Tuesday, April 21, 2015

**Data Forwarding Equations:**

1. EX hazard:

\*There are 2 inputs in the ALU and therefore we must make sure that the register value that is currently being updated is not being used in both of the A (Rs) and B (Rt) inputs of the ALU.

if (EX/MEM.RegWrite && (EX/MEM.RegisterRd != 0) && (EX/MEM.RegisterRd == ID/EX.RegisterRs) )

ForwardA = 10;

if (EX/MEM.RegWrite && (EX/MEM.RegisterRd != 0) && (EX/MEM.RegisterRd == ID/EX.RegisterRt) )

ForwardB = 10;

2. MEM hazard:

\*There are 2 inputs in the ALU and therefore we must make sure that the register value that is currently being updated is not being used in both of the A (Rs) and B (Rt) inputs of the ALU.

if(MEM/WB.RegWrite && (MEM/WB.RegisterRd !=0) && ! (EX/MEM.RegWrite && (EX/MEM.RegisterRd != 0) && (EX/MEM.RegisterRd == ID/EX.RegisterRt)) && (MEM/WB.RegisterRd == ID/EX.RegisterRs)

ForwardA = 01;

if(MEM/WB.RegWrite && (MEM/WB.RegisterRd !=0) && ! (EX/MEM.RegWrite && (EX/MEM.RegisterRd != 0) && (EX/MEM.RegisterRd == ID/EX.RegisterRt)) && (MEM/WB.RegisterRd == ID/EX.RegisterRt)

ForwardB = 01;

\*inside of the Data Forwarding block of code, we would make it so that the specified input to the ALU would equal the updated value of EX/MEM.RegisterRdData whenever the value of 10 was written to ForwardA or ForwardB. Otherwise the code would check to see if the MEM/WB.RegisterRd value was the same as the register value being updated. If this was the case, then we would update the specified input ALU value to the value of MEM/WB.RegisterRdData by writing a value of 01 to ForwardA or ForwardB. If neither one of these cases was true, then this means that there are no data hazards present and the ALU would just use the default ID/EX.RegisterRs and ID/EX.RegisterRt values as inputs.

**Pipeline Stall Implementation:**

\*Load is the only instruction that reads data memory, so we need to check if this control signal is asserted. After this, we need to check to see if the destination register field of the load in the EX stage matches either source register in the instruction in the ID stage. If they do match then we stall the system by 1 clock cycle.

if (ID/EX.MemRead && ((ID/EX.RegisterRt == IF/ID.RegisterRs) || (ID/EX.RegisterRt == IF/ID.RegisterRt)))

stall the pipeline

\*we stall the pipeline by inserting nop instructions (instructions that do no operations to change the state). This is achieved by changing the EX, MEM, and WB control signals of the ID/EX pipeline register to 0. These benign control values are percolated forward at each clock cycle with the proper effect: no registers or memories are written if the control values are all 0!

\*Note, we must also prevent the Program Counter value from changing / save its current value and prevent it from changing while the system is stalling...