Name ________________________

CE-1921-11 – Dr. Durant – Quiz 8
Spring 2016, Week 9

---

1. (2 points) *Describe* how pipelining *improves throughput*.
2. (2 points) *List* the five stages of instruction execution in ARM pipeline microarchitectures.
3. (2 points) *Justify* the *Harvard organization* in *pipelined* implementations.
4. (2 points) *Describe* the read-after-write hazard window for ARM pipeline microarchitectures if hazard-protection is not implemented.
5. (2 points) *List* the *three* principal techniques used to remove data or control hazards. Briefly define each technique.

Answers

1. In the SCP, much of the processor hardware sits idle at any given instant as data propagates through the various instruction stages (it has either completed its work and is outputting constants, or it is sitting idle waiting for inputs to arrive). A pipelined processor keeps each stage busy by marshalling multiple instructions through the hardware.
2. IF – ID – EX – MEM – WB ::: fetch – decode – execute – memory – write back
3. IF accesses instruction memory every cycle. MEM accesses data memory every cycle when a LDR/STR instruction is in the MEM stage. If there were a unified instruction/data memory, the design would be complicated since potentially two items would need to be read (or one read and one written) each clock cycle.
4. The hazard window is 2 cycles. The register data used at EX might be in the hazard window ahead 1 stage (MEM) or 2 stages (WB) and thus would not have been available in the register file when the instruction in EX was at its ID stage (in the previous clock cycle). (Recall that WB happens in the first half of the cycle and ID in the 2nd half – these must be separate in time since they both use the register file. If their order were reversed, the hazard window would increase to 3.)
5. The three techniques are
    a. Forwarding – data sent to ALU MUXes that wasn't available at ID
    b. Stalling – stopping an instruction from advancing through the pipeline for 1 (or more) clocks, inserting a bubble in the next stage
    c. Flushing – replacing an instruction in progress with a bubble, such as when it is discovered that a fetched instruction must not be executed due to a branch before it

Name ________________________

CE-1921-12 – Dr. Durant – Quiz 8
Spring 2016, Week 9

1. (2 points) *State the number* of instructions in flight every clock cycle when a basic ARM pipelines is full.
2. (2 points) *Describe* the purpose of interstage registers in pipeline microarchitecture.
3. (2 points) *List the three* types of pipeline hazards discussed in class *and give an instruction* that could cause each.
4. (2 points) *Describe* how the *hazard window* suggests *forwarding* paths for data hazard prevention.
5. (2 points) *Justify* advancing branch target circuitry into earlier pipeline stages such as IF or ID.

Answers

1. 5, 1 for each stage
2. Move the data through the pipeline at each start of a clock cycle; also moves the control signals that are needed by future stages in the pipeline.
3. The 3 types are
    a. Read after write – add r0, r1, r2; 1 or 2 instructions after r1 (or r2) was modified
    b. LDR – LDR r0,[r1,#4] followed by add r2,r0,r3
    c. Branch/control – b L0
4. The hazard window is 2 cycles – for longer windows between register write and use the data has been written back (WB) to the register file in time to be correctly decoded. For a 1-cycle window, the needed register value can be forwarded from the MEM stage; for a 2-cycle window, the needed register value can be forwarded from the WB stage [note: it is being written back as this happens, but it is too late to decode since the forwarding is to the EX stage, not the ID stage].
5. Our best pipelined processor designed in class resolved branches (both determined the branch target made the decision whether to write that target to the PC) in the EX stage, meaning that when a branch is taken, 2 instructions (those in IF and ID) need to be flushed, reducing performance due to 2 bubbles being inserted into the pipeline (increasing the CPI numerator by 2 without affecting the denominator). Resolving branches at IF or ID, respectively, takes more hardware and increases design complexity, but only increases the CPI numerator by 0 or 1 when a branch is taken.