**NAME**

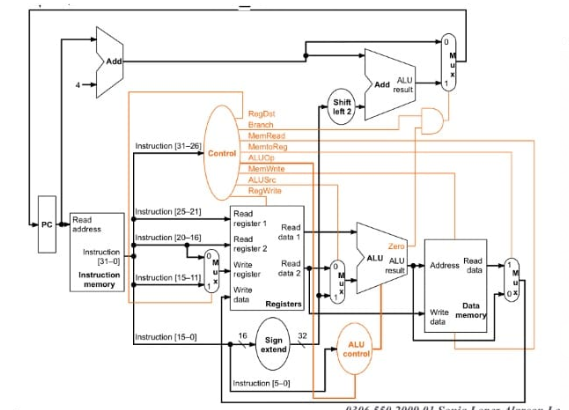
**COLLEGE NUMBER**

**QUESTION ONE**

**Section 1**

In this question, you are adding addi I-type instruction, which is in the format of “addi Rd, Rs1, imm”. The operation is to perform Reg[Rd] = Reg[Rs1] + Imm. Highlight the datapath used for this instruction based on the datapath figure shown below. (10 points)

Hint: the instruction is very similar to LW, except it needs to write the ALU output to the register, and you can use LW datapath as reference for this question. Check slides about how to highlight data path for instructions.



**Section 2**

Your work is to fill in the provided Excel sheet the values of the data path that are relevant to the instruction CPU is executing, and the setting of control signals of each type of instruction. Datapaths are labeled as in the following diagram. If you need references, book chapter 4.4 and 4.5 provide detailed description about how each type of instruction is executed, what datapath each uses and what control signals each instruction set or reset. Some of the pictures of the textbook are already copied to the answer sheet to help you look up. Check the comments of some cell (if they have comments) for details how the value is calculated. To simplify answering with the sheet, we assume that the execution of the instruction does not change the actual value in the register files and memory.

**Section 3**

add xii, x6,-2   beqxn,xii,     exit

add i x12,      x10,1 lw        x12

lix10,x10,0     lwx8,q(x12)     x10

lix10,0         lwx7,1(x10)     x10

lix10,0         add:x12,x10,1   x10

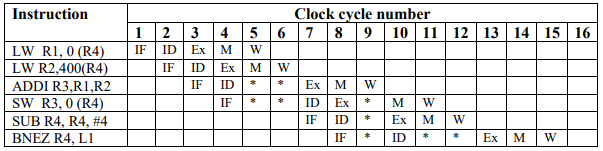
lwx7,1(x10)     add x9,x7,x8    x7

lwx8,a(x12)     add x9,x7,x8    x8

lix10,0         lw,x9,a(x10)    x10

--The Rows are load-use. This means that the registers are

first loaded and the used in next instruction

****

During the 5th and 8th clock cycles, there are two time slots that are not utilised (stalls). We can fill the 5-th c.c. slot by rescheduling the SUB instruction after the LW R2. To occupy the 8th c.c. time slot, the SW instruction might be moved to the end of the loop (after BNEZ). There will be no stalls now, and the minimum number of clock cycles required for segment processing will be 6 c.c. When compared to a non-pipelined processor, the highest speedup is = = 3005 / (1+ 6 x 100) = 5 times. During task segment execution, all stages of the 5-stage pipeline are always busy (no stalls).The delay of data in the register R2 for the ADDI causes two stall cycles (c.c. # 5 and 6)

2. The ID stage circuits are busy for ADDI and only become accessible on the 7th c.c., resulting in the same stall cycles in the ID stage for the SW instruction.

3. Because the IF stage is occupied with SW instructions, SUB can only begin on the 8th c.c.

4. The content of R3 (for SW) is not available, causing a c.c. stall in the pipeline.

For SW instructions, however, the "Ex" stage can be used.

As a result,

Because the address in memory is computed during the "Ex" stage (just for Load or Store instructions), this is conceivable.

5. The delay in updating the R4 causes two stall cycles in BNEZ (c.c. # 11 and 12).

Only after 12 c.c. does new R4 content become available.

As a result, the material of PC is updated on a regular basis.

**Section 4**

Draw the 5-stage pipeline execution of the first TWO iterations using stage labels (IF, ID, EXE, MEM, and WB) with no any data forwarding in the CPU. All RAW dependencies between consecutive two instructions (including AL-Use and Load-use) cause 2 cycle delay. BEQ cause

add xii, x6,-2   beqxn,xii,     exit

add i x12,      x10,1 lw        x12

lix10,x10,0     lwx8,q(x12)     x10

lix10,0         lwx7,1(x10)     x10

lix10,0         add:x12,x10,1   x10

lwx7,1(x10)     add x9,x7,x8    x7

lwx8,a(x12)     add x9,x7,x8    x8

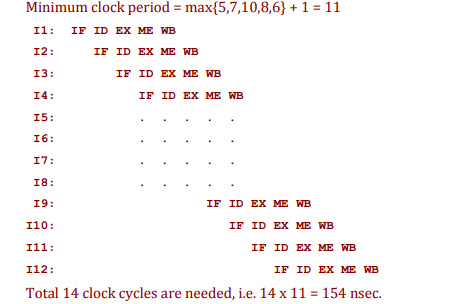
lix10,0         lw,x9,a(x10)    x10

--The Rows are load-use. This means that the registers are

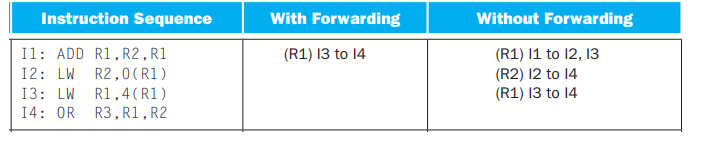
first loaded and the used in next instruction

A pipeline with N stages could increase throughput by N times, but » each step must take the same amount of time » each stage must always have work to perform » there may be some overhead to implement.We don't drive outside of certain parameters.

Draw the 5-stage pipeline execution of the first TWO iterations using stage labels on the CPU but with fully data forwarding. With forwarding, the AL-Use RAW dependencies between consecutive two instructions cause 0 cycle delay. The Load-use dependency between consecutive two instructions cause 1 cycle delay. BEQ still has two cycle delay for issuing next instruction.



Only RAW dependences can become data hazards. With forwarding, only RAW dependences from a load to the very next instruction become hazards. Without Only RAW dependences can become data hazards. With forwarding, only RAW dependences from a load

****

For the CPU with fully data forwarding, rearrange instructions using the techniques we discussed in the Chapter 4 lecture to eliminate the stall(s) from load-use hazard. All the load-use stall cycles should be completely eliminated, and then draw the 5-stage pipeline execution of the first TWO iterations using stage labels. When you do reschedule, you are allowed to change instruction to make sure the code is executed correctly. Hints, you are allowed to have “SW, a+/-ConstantOffset (base)” format, e.g. “sw x11 a-4(x10)” meaning to store x11 to address “a-4+x10”

