CS420/520 Homework Assignment: Pipelining

Total: 100 points.

6.2 [10]: Using a drawing similar to the Figure 6.8 below, show the forwarding paths needed to execute the following three instructions:

Add $2, $3, $4
Add $4, $5, $6
Add $5, $3, $4

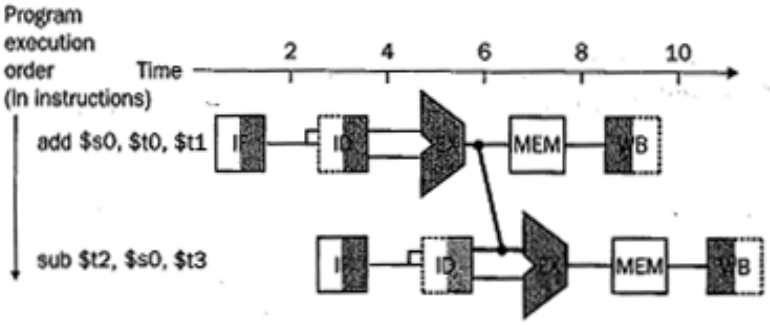


FIGURE 6.8 Graphical representation of forwarding. The connection shows the forwarding path from the output of the EX stage of add to the input of the EX stage for sub, replacing the value from register $s0 read in the second stage of sub.

6.3 [5] How could we modify the following code to make use of a Delayed Branch Slot (DBS)?

```
Loop:  lw $2, 100($3)
       Addi $3, $3, 4
       Beq $3, $4, Loop
```

* Note that reordering the code sequence aside, you may have to modify some instruction(s).

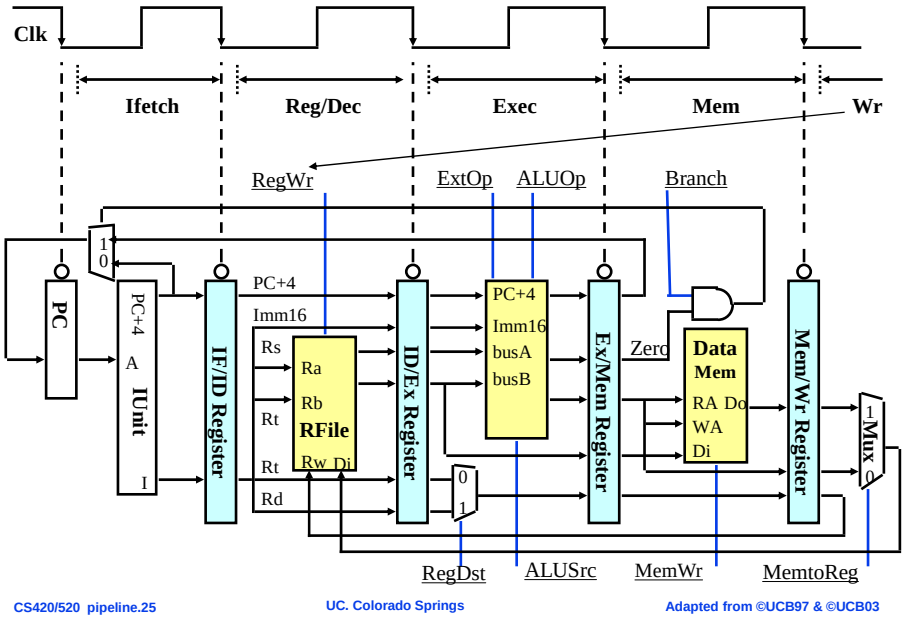
6.4 [10] First please identify all of the data dependencies in the following code. Then, assume the simultaneous reading and writing, which dependencies are data hazards that will be resolved via data forwarding? Please draw a diagram to show the backward data dependencies as that shown in the lecture notes.

```
Add $2, $5, $4
Add $4, $2, $5
Sw $5, 100 ($2)
Add $3, $2, $4
```

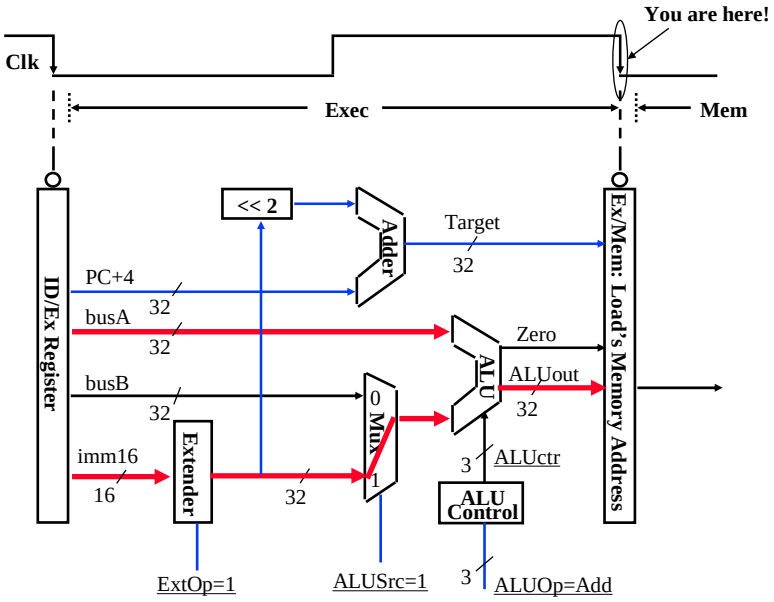
6.5 [15] For each pipeline register in the following diagram (also refer to the detailed view of the execution unit), label each portion of the pipeline latch register with the name of the value that is loaded into the register and determine the length of each field in bits.

For example: the IF/ID pipeline register holds *PC+4* (32 bits) and the *instruction* (32 bits) for a total of 64 bits.

A Pipelined Datapath



A Detail View of the Execution Unit (Integrated)



6.7 [10] Using the Figure 6.25 below as a guide, use colored pens or markers to show which portions of the datapath are actually used in each of the five stages of the ***add*** instruction. We suggest that you use five copies of the figure to do so.

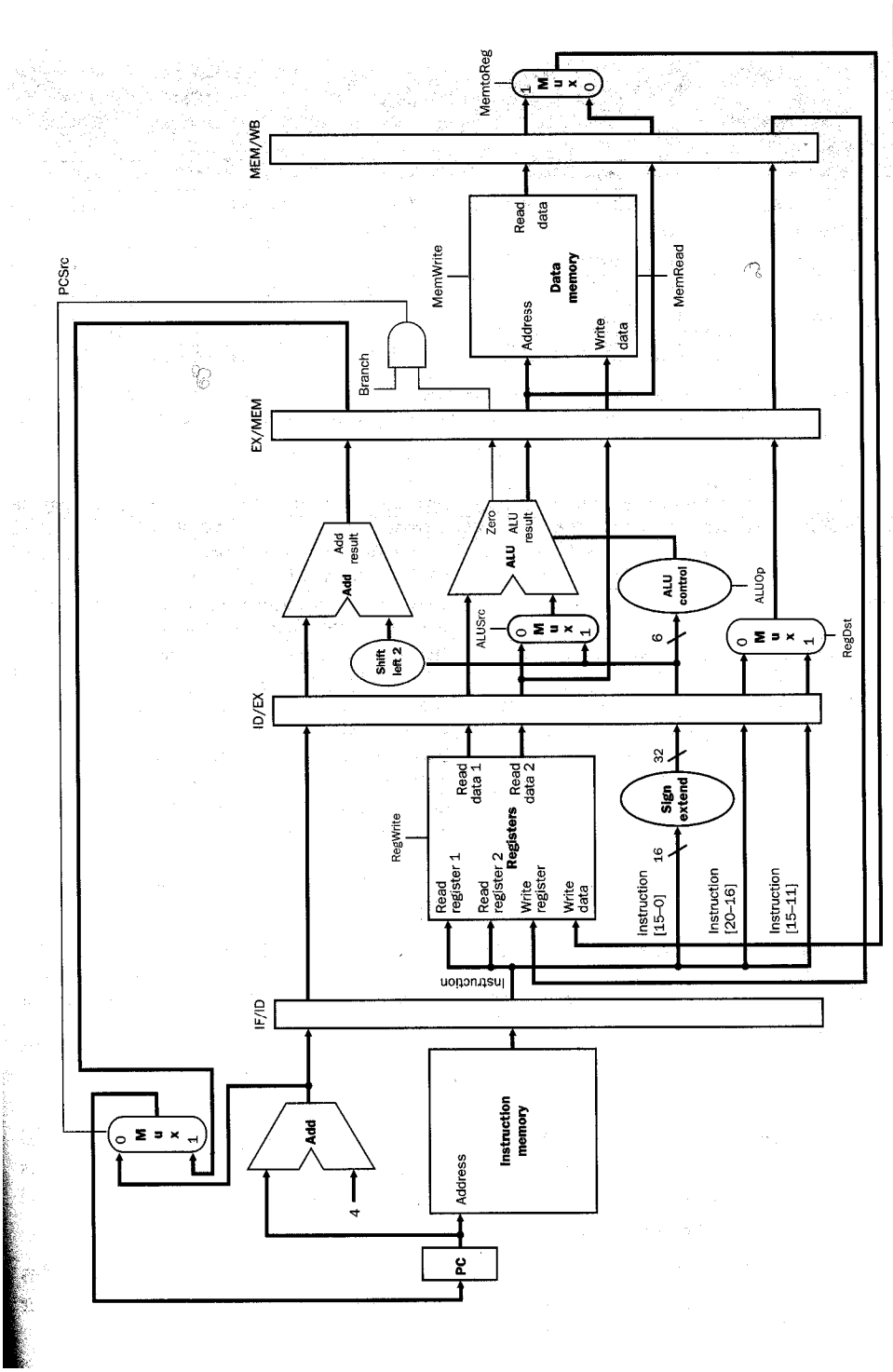


FIGURE 6.25 The pipelined datapath of Figure 6.18 with the control signals identified. This datapath borrows the control logic for PC source, register destination number, and ALU control from Chapter 5. Note that we now need the 6-bit funct field (function code) of the instruction in the EX stage as input to ALU control, so these bits must also be included in the ID/EX pipeline register. Recall that these 6 bits are also the 6 least significant bits of the immediate field in the instruction, so the ID/EX pipeline register can supply them from the immediate field since sign extension leaves these bits unchanged.

6.11 [5] Consider executing the following code on the pipelined datapath of Figure below:

```
add $1, $2, $3
add $4, $5, $6
add $7, $8, $9
add $10, $11, $12
add $13, $14, $15
```

At the end of the fifth cycle of execution, which registers are being read from the Register File and which register is written to the Register File?

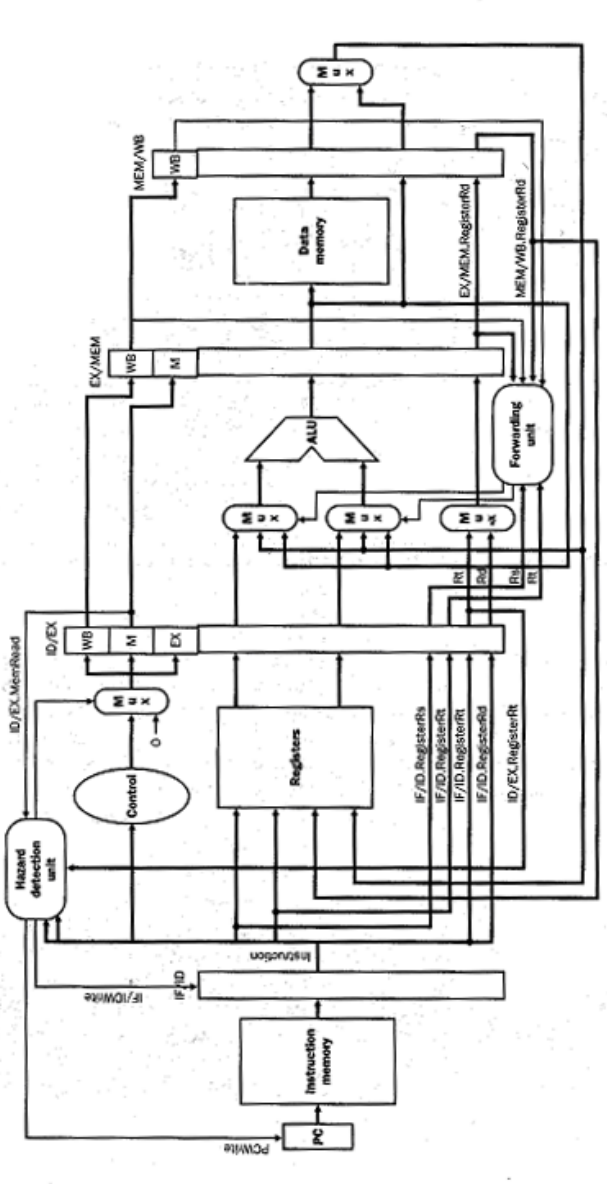


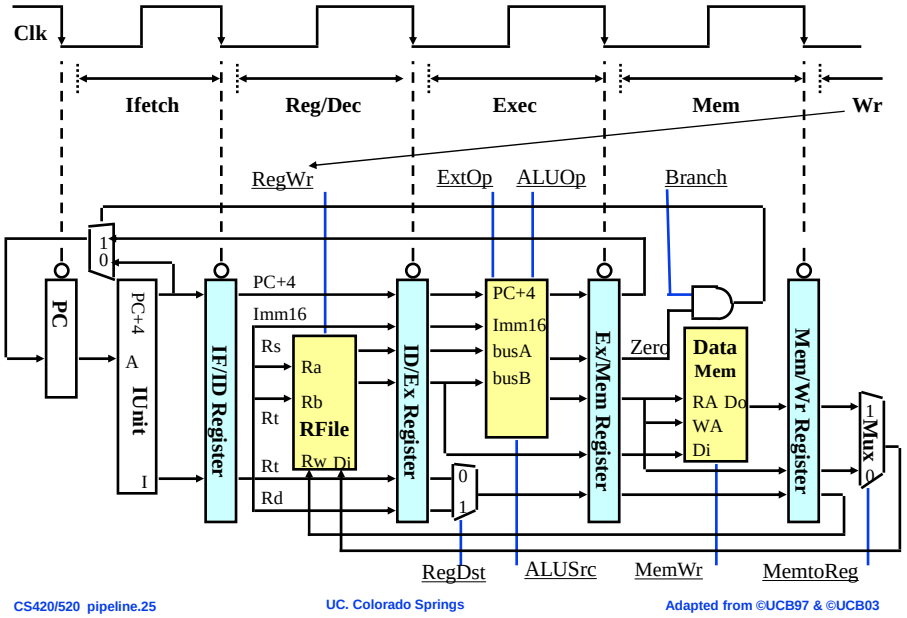
FIGURE 6.46 Pipelined control overview, showing the two multiplexors for forwarding, the hazard detection unit, and the forwarding unit. Although the ID and EX stages have been simplified—the sign-extended immediate and branch logic are missing—this drawing gives the essence of the forwarding hardware requirements.

6.12 [10] With regard to the program in Exercise 6.11, explain what the forwarding unit is doing during the fifth cycle of execution, if any comparisons of register numbers are being made, mention them. Refer to “Load Interlock Control” on Pipeline II Power Point Slides for the clue.

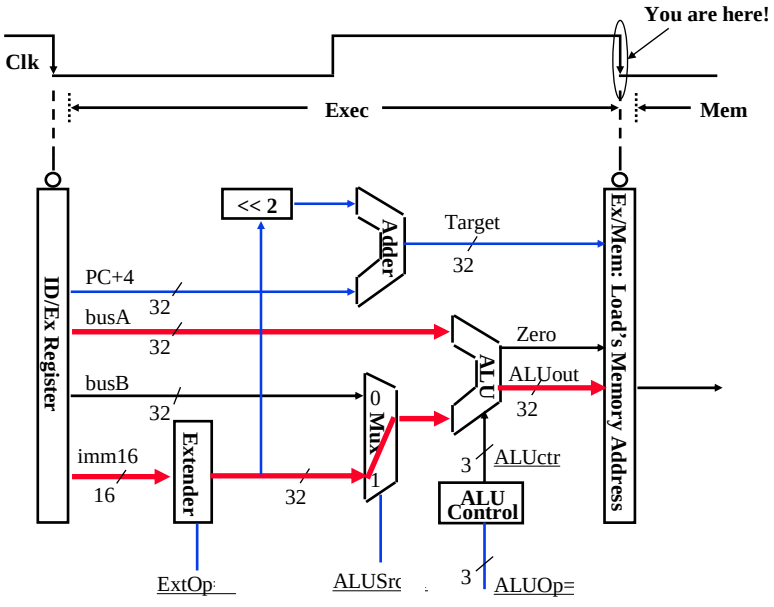
6.13 [10] Following five MIPS instructions go into the pipelined datapath below (the first one, Load, goes into the datapath in cycle 1). Set the control signal values in the end of clock cycle 5.

Load $1, $(0)
Add $3, $4, $5
SW $6, $7(12)
Beq $8, $9, 100
Add $11, $12, $13

A Pipelined Datapath



A Detail View of the Execution Unit (Integrated)



6. 15 [25] Use the following code fragment

```
Loop:  LD      R1, 0(R2)      ;load R1 from memory at address 0+R2
       DADDI   R1, R1, #1     ; R1 = R1 + 1
       SD      R1, 0(R2)      ; store R1 to memory at address 0+R2
       DADDI   R2, R2, #4     ; R2 = R2 + 4
       DSUB    R4, R3, R2     ; R4 = R3 – R2
       BNEZ    R4, Loop       ; branch to Loop if R4 != 0. Note BNEZ
                              ; outcomes and targets are not known until
                              ; the end of the execute stage.
```

Assume that the initial value of R3 is R2 + 396. Assume using the classic five-stage integer pipeline and assume that all memory accesses take 1 clock cycle.

Clue: using the following pipeline timing chart for your solution:

```
                      1  2  3  4  5  6  7  8  9 . . .
LD   R1, 0(R2)        F  D  X  M  W
DADDI R1, R1, #1         F  s  s  D  X  M  W
```

F = IF (instruction fetch stage)
D = ID/RF (instruction decoding and register fetch stage)
X = Exec (execution stage)
M = Mem (memory access stage, be reading or writing)
W = WriteBack (write the register)

a. Show the timing of this instruction sequence for the RISC pipeline *without* any forwarding or bypassing hardware but assuming simultaneous register reading and writing in the same clock cycle. Furthermore, assume that the branch is handled by *flushing the pipeline*. Please draw a pipeline timing chart for the instruction sequence, like the example above. And, if all memory references take 1 cycle, how many cycles does this loop take to execute?

b. Show the timing of this instruction sequence for the RISC pipeline *with* forwarding and bypassing hardware, assuming simultaneous register reading and writing in the same clock cycle. Furthermore, assume that the branch is handled by *predicting it as not taken*. Please draw a pipeline timing chart for the instruction sequence, like the example above. And, if all memory references take 1 cycle, how many cycles does this loop take to execute?