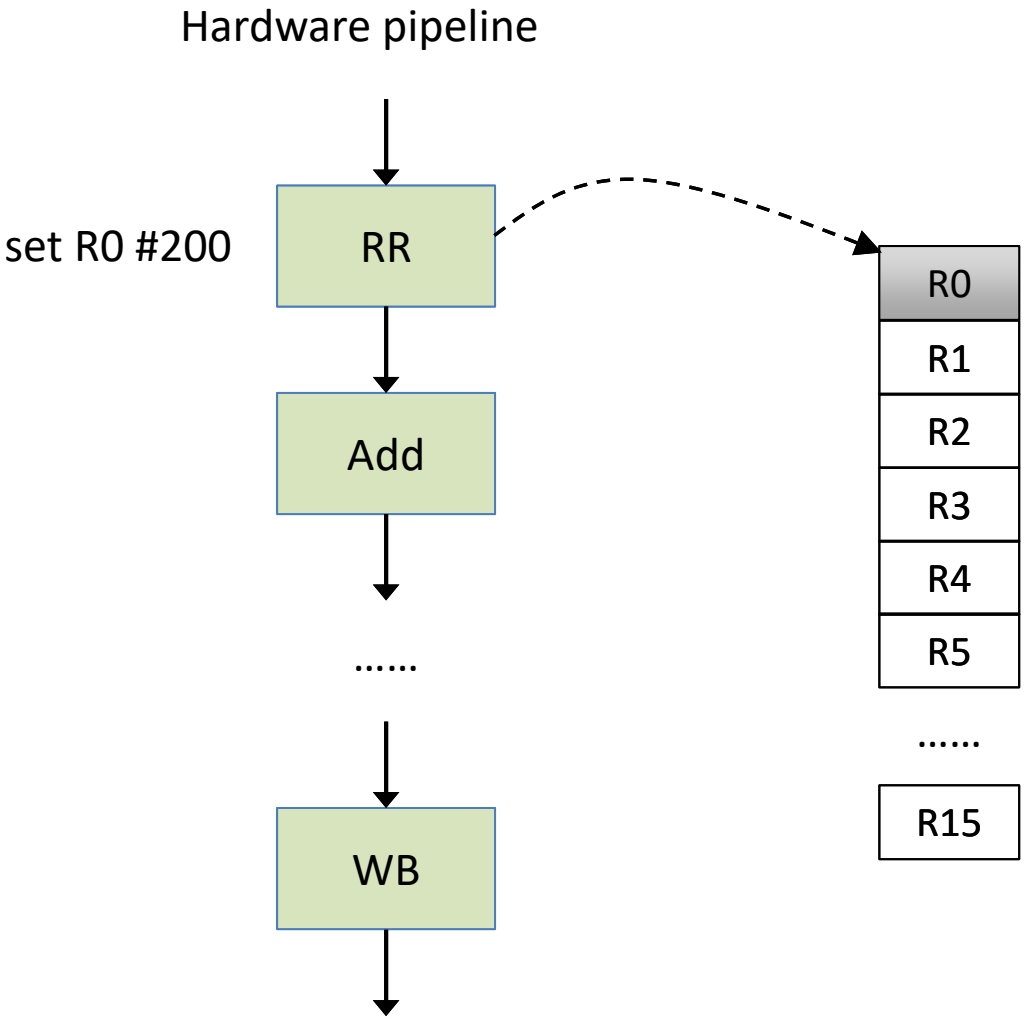
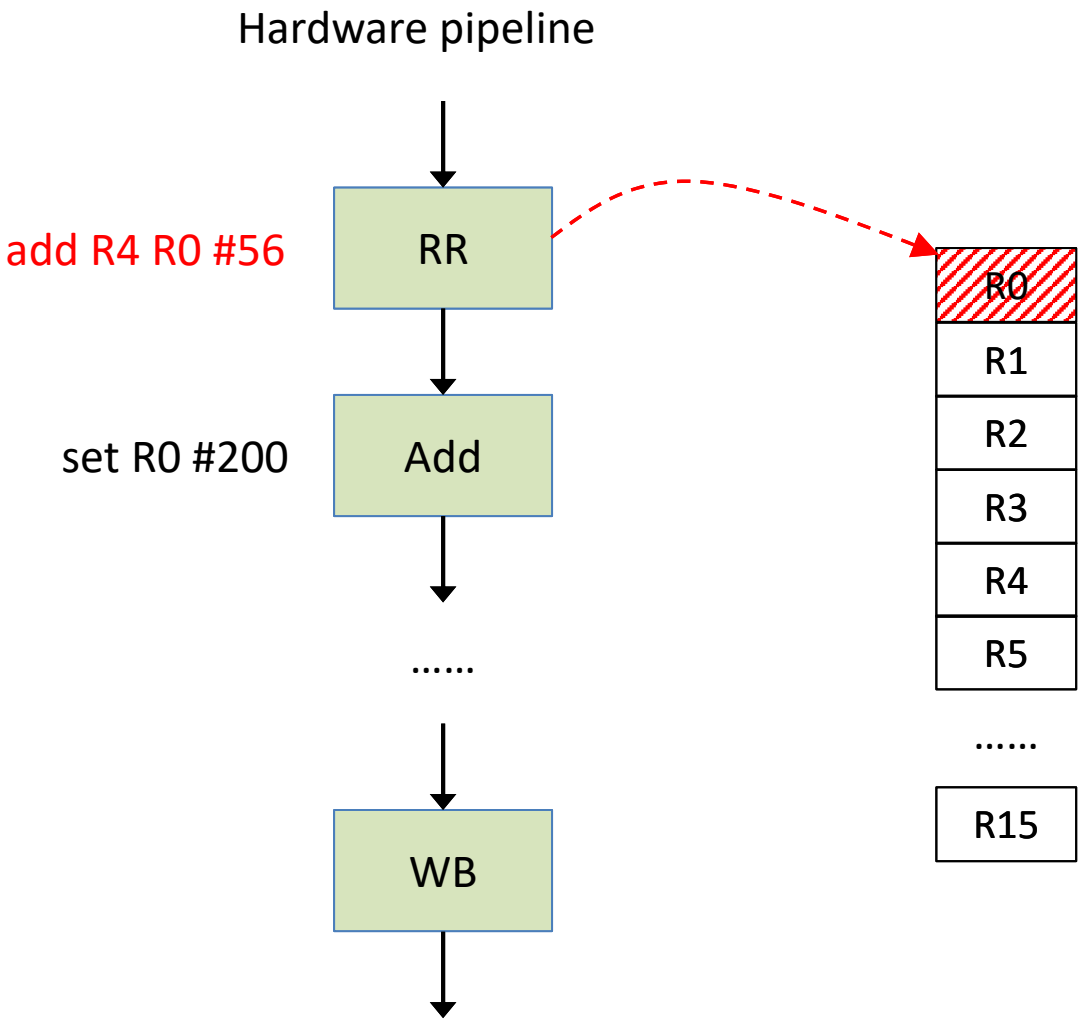
CS 520 Computer Architecture

Project #2

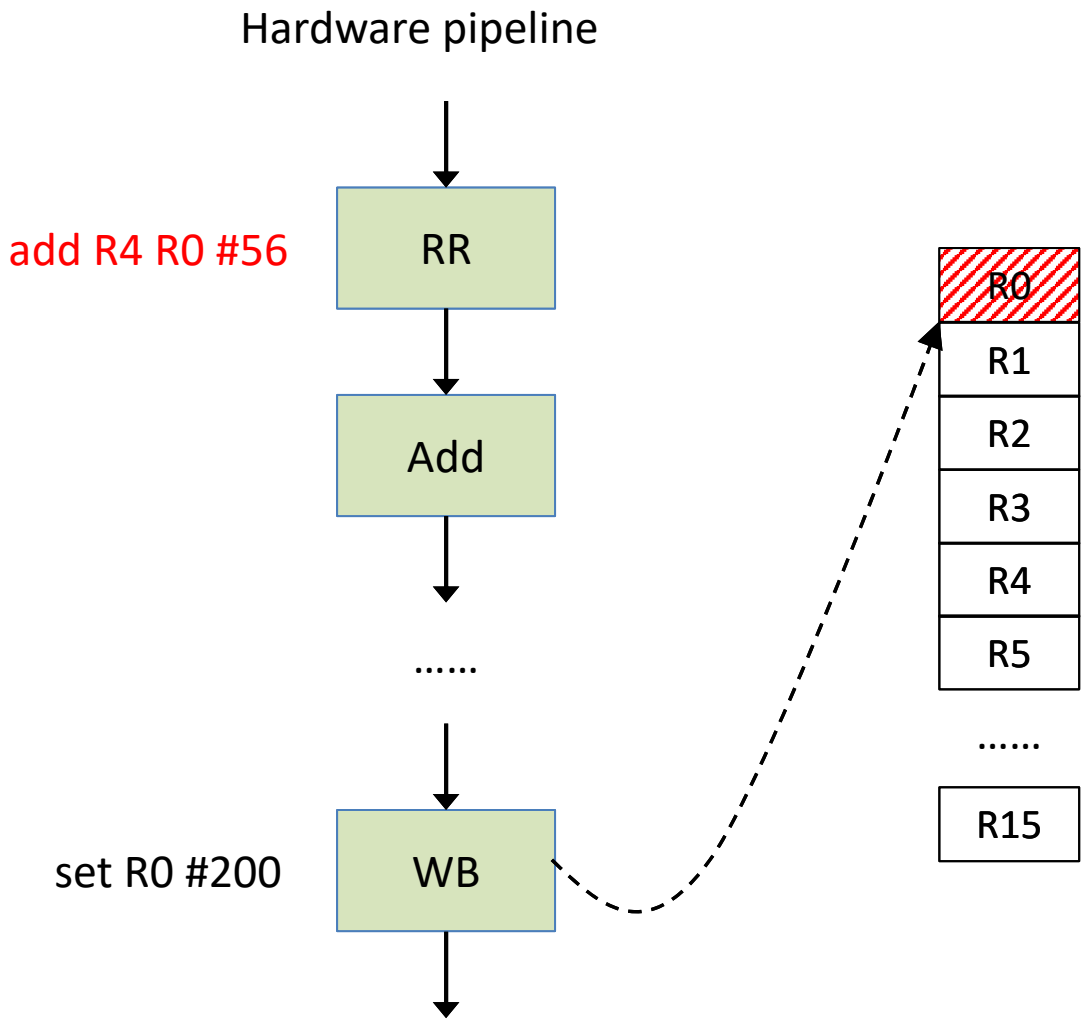
Data Hazard in Pipeline (1)



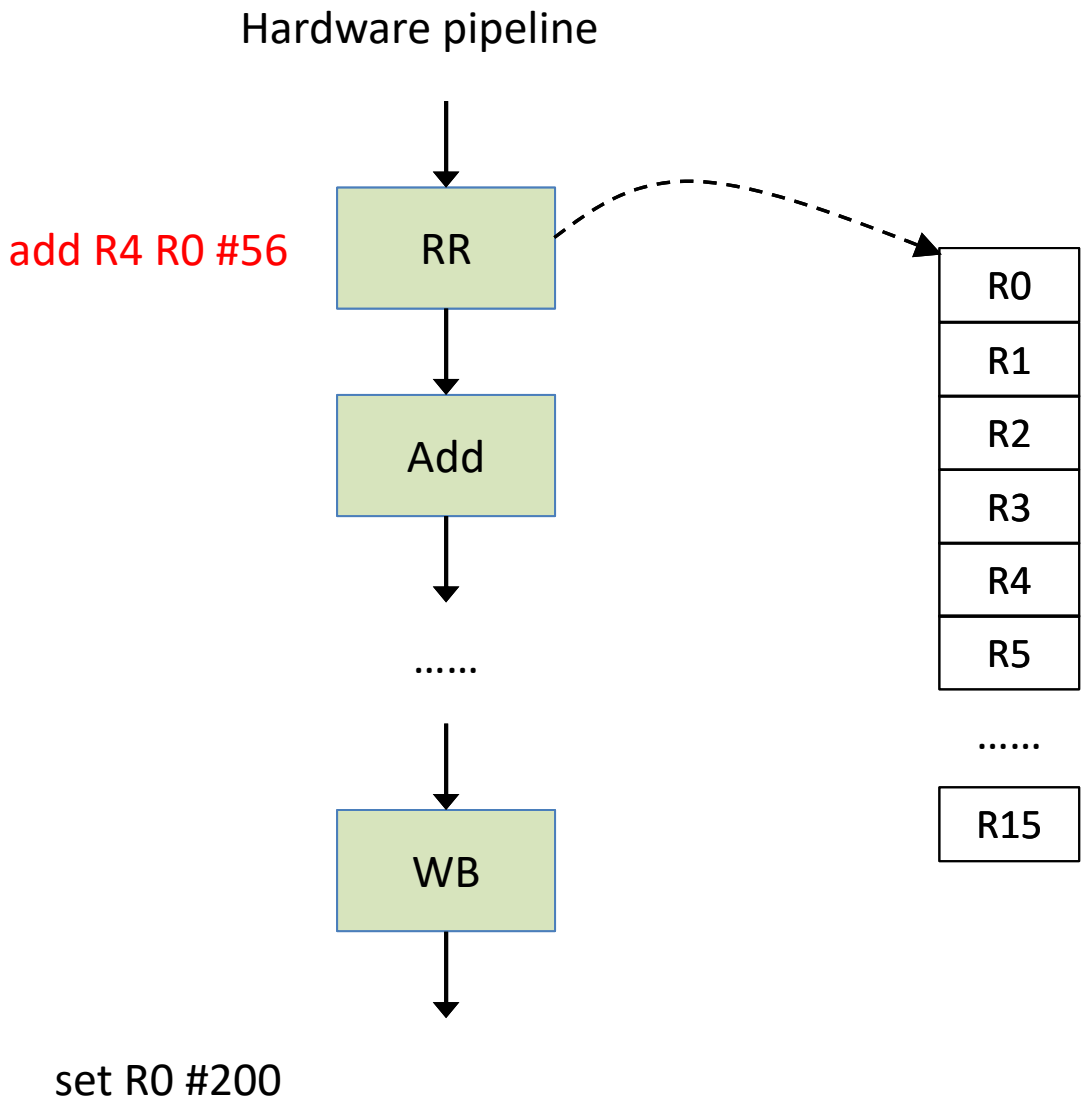
Data Hazard in Pipeline (2)



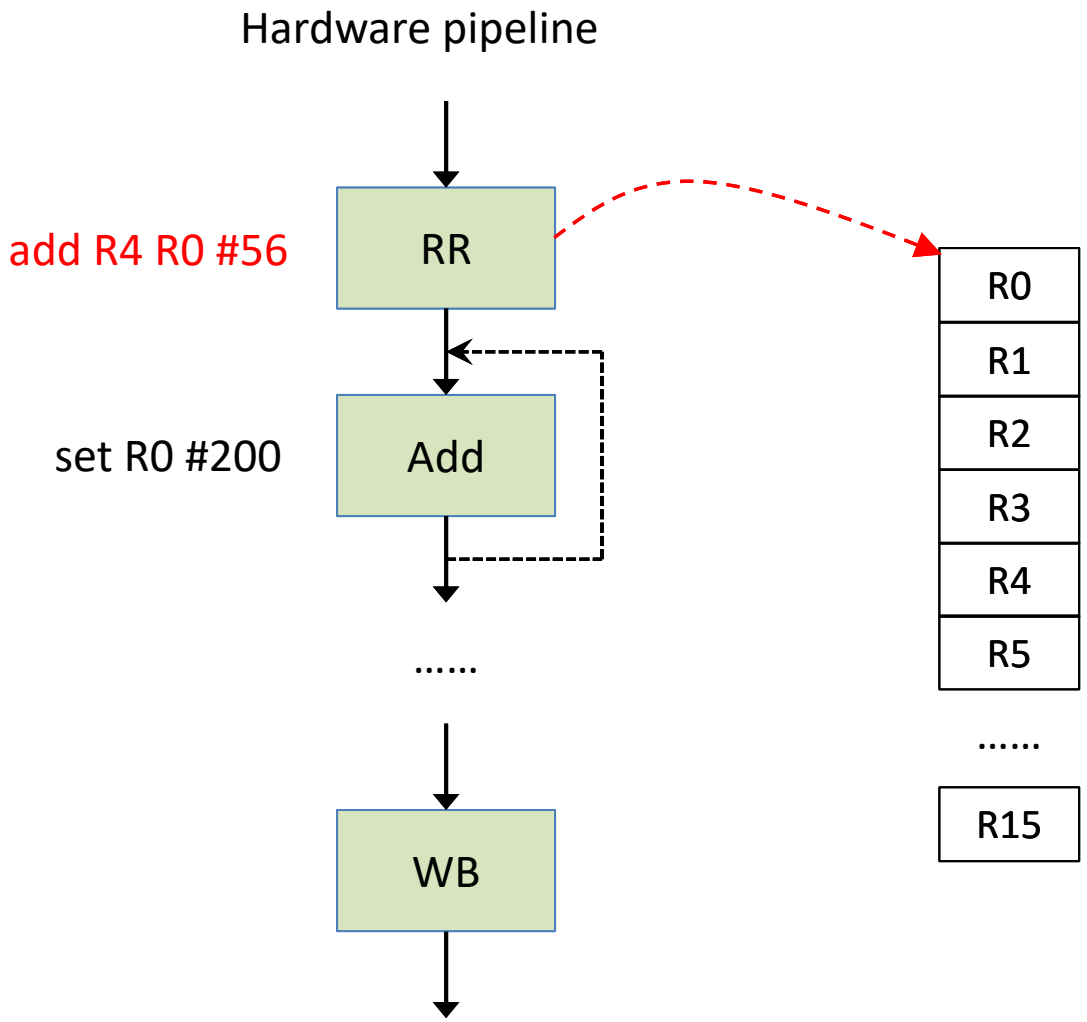
Data Hazard in Pipeline (3)



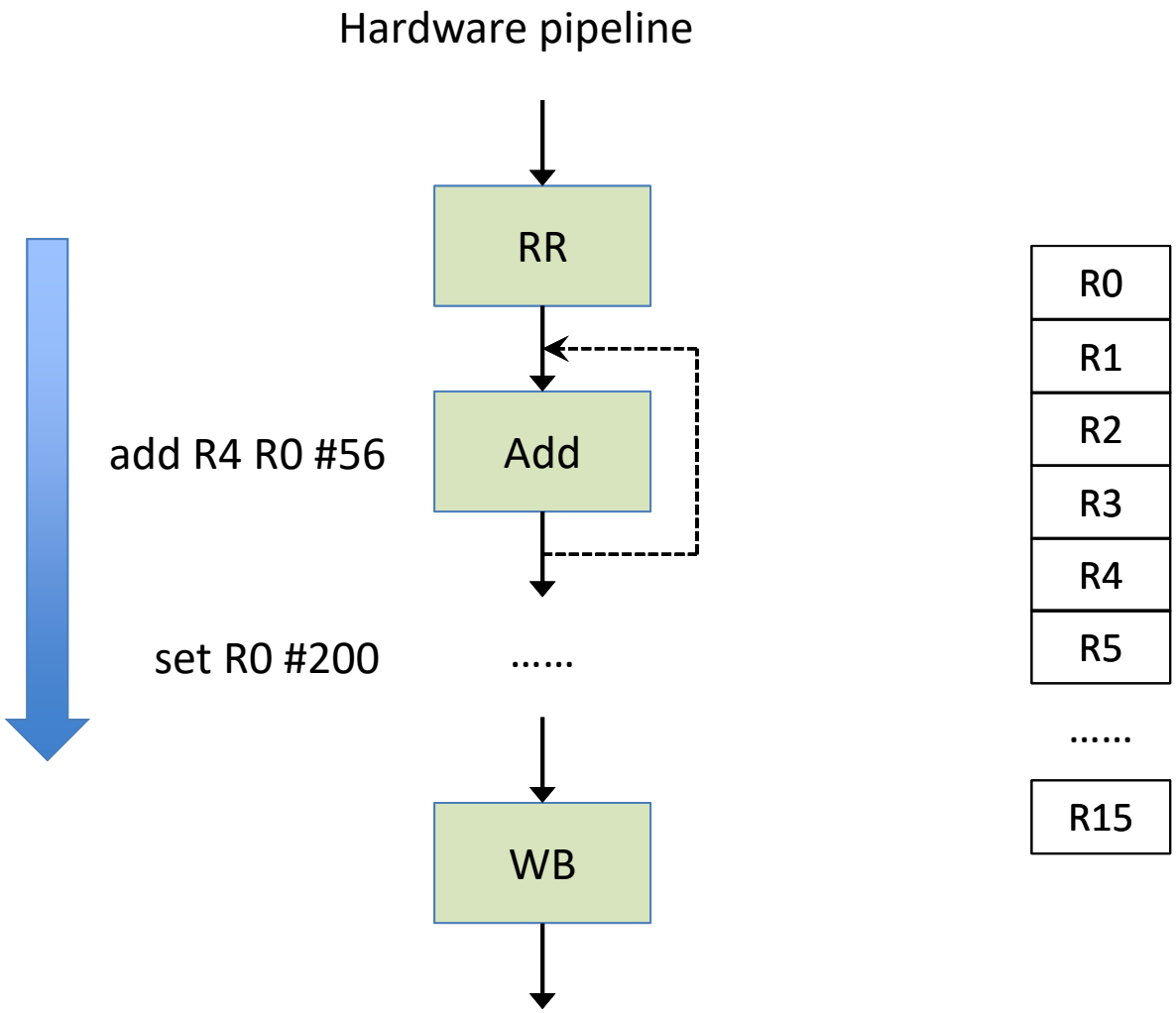
Data Hazard in Pipeline (4)



Data Hazard with Forwarding (1)



Data Hazard with Forwarding (2)



Output (1) – name_result.txt

```
--------------------------------
REG[15]   |   Value=4556
--------------------------------
================================

================================

=============== STATE OF ARCHITECTURAL REGISTER FILE ==========

--------------------------------
REG[ 0]   |   Value=-2720
--------------------------------
REG[ 1]   |   Value=1752
--------------------------------
REG[ 2]   |   Value=7039656
--------------------------------
REG[ 3]   |   Value=6072
--------------------------------
REG[ 4]   |   Value=2834
--------------------------------
REG[ 5]   |   Value=37
--------------------------------
REG[ 6]   |   Value=4446
--------------------------------
REG[ 7]   |   Value=12911704
--------------------------------
REG[ 8]   |   Value=39
--------------------------------
REG[ 9]   |   Value=4420
--------------------------------
REG[10]   |   Value=7492
--------------------------------
REG[11]   |   Value=90
--------------------------------
REG[12]   |   Value=5456
--------------------------------
REG[13]   |   Value=2484
--------------------------------
REG[14]   |   Value=0
--------------------------------
REG[15]   |   Value=4556
--------------------------------
================================

Stalled cycles due to data hazard: 0
Total execution cycles: 33
Total instruction simulated: 23
IPC: 0.696970
```

- name_result.txt shows the output results

Output (2) – Printing Format

```
/*
 * This function prints the content of the registers.
 */
void
print_registers(CPU *cpu){


    printf("================================\n");
    printf("--------------------------------\n");
    for (int reg=0; reg<REG_COUNT; reg++) {
        printf("REG[%2d]   |   Value=%d  \n",reg,cpu->regs[reg].value);
        printf("--------------------------------\n");
    }
    printf("================================\n\n");
}

/*
 *  CPU CPU simulation loop
 */
int
CPU_run(CPU* cpu)
{

    print_registers(cpu);

    printf("Stalled cycles due to data hazard: \n");
    printf("Total execution cycles: \n");
    printf("Total instruction simulated:\n" );
    printf("IPC: \n");


    return 0;
}
```

- cpu.c already contains codes for printing

Logs– name_pipeline.txt

```
================================
Clock Cycle #: 1
--------------------------------
IF              : 0000 set R6 #8892
================================
Clock Cycle #: 2
--------------------------------
ID              : 0000 set R6 #8892
IF              : 0004 set R7 #6172
================================
Clock Cycle #: 3
--------------------------------
IA              : 0000 set R6 #8892
ID              : 0004 set R7 #6172
IF              : 0008 set R1 #1752
================================
Clock Cycle #: 4
--------------------------------
RR              : 0000 set R6 #8892
IA              : 0004 set R7 #6172
ID              : 0008 set R1 #1752
IF              : 0012 set R4 #2834
================================
Clock Cycle #: 5
--------------------------------
ADD             : 0000 set R6 #8892
RR              : 0004 set R7 #6172
IA              : 0008 set R1 #1752
ID              : 0012 set R4 #2834
IF              : 0016 ld R8 #31328
```

- name_pipeline.txt shows all the information including the register status at each cycle