Faculty of Media Engineering and Technology
Dept. of Computer Science and Engineering
Dr. Milad Ghantous

CSEN 702: Microprocessors
Winter 2023

Practice assignment 5

Exercise 1

The following loop is the so-called DAXPY loop (double-precision aX + Y) and is the central operation in Gaussian elimination. The following code implements the DAXPY operation, Y = aX + Y, for a vector length 100.
Initially, R1 is set to the base address of array X and R2 is set to the base address of Y.

```
      DADDIU   R4,R1,#800  ;  R1 = upper bound for X
foo:  L.D      F2,0(R1)    ; (F2) = X(i)
      MUL.D    F4,F2,F0    ; (F4) = a*X(i)
      L.D      F6,0(R2)    ; (F6) = Y(i)
      ADD.D    F6,F4,F6    ; (F6) = a*X(i) + Y(i)
      S.D      F6,0(R2)    ; Y(i) = a*X(i) + Y(i)
      DADDIU   R1,R1,#8    ; increment X index
      DADDIU   R2,R2,#8    ; increment Y index
      DSLTU    R3,R1,R4    ; test: continue loop?
      BNEZ     R3,foo      ; loop if needed
```

- Assume the functional unit latencies as shown in the table below.
- Assume a one cycle delayed branch that resolves in the ID stage. One cycle delayed branches have 1 stall inserted after them.
- Assume that results are fully bypassed. (The table already shows the latency needed when bypassing is applied)

Instruction producing result	Instruction using result	Latency in clock cycles
FP multiply	FP ALU op	5
FP add	FP ALU op	3
FP multiply	FP store	4
FP add	FP store	3
Integer operations and all loads	Any	1

1.1) Show the unscheduled loop and calculate the number of cycles it needs.
1.2) Show the scheduled loop and calculate number of cycles it needs. How much improvement over the unscheduled code?
1.3) Unroll the loop 3 times without any scheduling and compute the number of cycles needed per 1 iteration. Compare with the rest.
1.4) Unroll the loop 3 times with scheduling and compute the number of cycles needed per 1 iteration. Compare with the rest.

Exercise 2

The loop iterations N might not be divisible by the unrolling factor K (the number of iterations we enroll the loop). Suggest a scheme to organize that.

Exercise 3

Consider the following code:

```
for (i=0; i<100; i=i+1)
{
  A[i+1] = A[i] + C[i];     /* S1 */
  B[i+1] = B[i] + A[i+1];   /* S2 */
}
```

Is there any loop-carried dependency? Can we get rid of it?

Exercise 4

Consider the following code:

```
for(i=0; i<100; i++) {

 A[i] = B[i]++;
 C[i] = A[i]*5;
 B[i] = B[i]/5;
 A[i] = D[i];

}

for(i=0; i<100; i++) {
 B[i]=1;
}
```

Mention all the dependencies types found in the code and suggest a renaming scheme to solve/avoid them to the best of your ability.