Homework Assignment 3

CS/ECE 6810: Computer Architecture

February 14, 2018

Instruction Level Parallelism

Due Date: 2/25/2018

(65 points)

Important Notes:

- Solutions turned in must be your own. Please, mention references (if any) at the end of each question. Please refrain from cheating.
- All solutions must be accompanied by the equations used/logic/intermediate steps. Writing only the final answer will receive **zero** credit.
- All units must be mentioned wherever required.
- Late submissions (**after 11:59 pm on 02/25/2018**) will not be accepted
- All submissions must be in PDF only. Scanned copies of handwritten solutions will not be accepted as a valid submission.

Scalar Processors: For the following questions, consider a basic in-order pipeline with bypassing (one instruction in each pipeline stage in any cycle). The pipeline has been extended to handle FP ADD and FP MULT. Assume the following delays between dependent instructions:

(a) Load feeding any instruction: 1 stall cycle
(b) FP MULT/ADD feeding store: 4 stall cycles
(c) Integer ADD feeding a branch: 1 stall cycle
(d) A conditional branch has 1 delay slot (an instruction is fetched in the cycle after the branch without knowing the outcome of the branch and is executed to completion)

1. **Software Optimization:** Below is the source code and default assembly code for a loop.

Source code

```
for (i=1000; i>0; i--) {
   x[i] = z[i] * y[i];
}
```

Assembly code

```
Loop: L.D F1, 0(R1)
      L.D F2, 0(R2)
      MUL F3, F1, F2
      S.D F3, 0(R3)
      DADDUI R1, R1, #-8
      DADDUI R2, R2, #-8
      DADDUI R3, R3, #-8
      BNE R1, R4, Loop
      NOP
```

i. Show an optimized schedule for this loop without resorting to unrolling.**(10 points)**

Answer:

```
Loop:  LD F1, 0(R1)
       LD F2, 0(R2)
       DADDUI R1, R1, #-8
       MUL F3, F1, F2
       DADDUI R2, R2, #-8
       DADDUI R3, R3, #-8
       Stall
       BNE R1, R4, Loop
       SD F3, 8(R3)
```

ii. Unroll the loop twice and schedule instructions to minimize stalls.**(10 points)**

Answer(s):

Loop: LD F1, 0(R1)
LD F2, 0(R2)
LD F4, -8(R1)
MUL F3, F1, F2
LD F5, -8(R2)
DADDUI R1, R1, #-16
MUL F6, F4, F5
DADDUI R2, R2, #-16
SD F3, 0(R3)
DADDUI R3, R3, #-16
BNE R1, R4, Loop
SD F6, 8(R3)
	Loop: LD F1, 0(R1)
	LD F2, 0(R2)
	LD F4, -8(R1)
	LD F5, -8(R2)
	MUL F3, F1, F2
	MUL F6, F4, F5
	DADDUI R1, R1, #-16
	DADDUI R2, R2, #-16
	DADDUI R3, R3, #-16
	SD F3, 16(R3)
	BNE R1, R4, Loop
	SD F6, 8(R3)

iii. Show the software pipelined version of this code. There is no need to show the prologue and epilogue.**(10 points)**

0	LD	LD	X	MUL	X	X	X	X	SD								
-8		LD	LD	X	MUL	X	X	X	X	SD							
-16			LD	LD	X	MUL	X	X	X	X	SD						
-24				LD	LD	X	MUL	X	X	X	X	SD					
-32					LD	LD	X	MUL	X	X	X	X	SD				
-40						LD	LD	X	MUL	X	X	X	X	SD			
-48							LD	LD	X	MUL	X	X	X	X	SD		
-56								LD	LD	X	MUL	X	X	X	X	SD	
-64									LD	LD	X	MUL	X	X	X	X	SD

```
        Prologue
Loop:   SD F3, 0(R1)
        MUL F3, F1, F2
        LD F2, -56(R1)
        LD F1, -64(R1)
        DADDUI R1, R1, #-8
        DADDUI R2, R2, #-8
        DADDUI R3, R3, #-8
        BNE R1, R4, Loop
        Epilogue
```

2. **Software Optimization:** Below is the source code and default assembly code for a loop.

Source code

```
for (i=1000;i>0;i--) {
x[i] = y[i] + z[i]*w[i];
}
```

Assembly code

```
Loop: L.D F2, 0(R2)
L.D F3, 0(R3)
MULT.D F1, F2, F3
L.D F4, 0(R4)
ADD.D F5, F4, F1
S.D F5, 0(R5)
DADDUI R2, R2, #-8
DADDUI R3, R3, #-8
DADDUI R4, R4, #-8
DADDUI R5, R5, #-8
BNE R2, R1, Loop
NOP
```

Show an optimized schedule for this loop without resorting to unrolling. **(10 points)**

Answer:

```
Loop:  LD F2, 0(R2)
       LD F3, 0(R3)
       LD F4, 0(R4)
       MULT.D F1, F2, F3
       ADD.D F5, F4, F1
       DADDUI R2, R2, #-8
       DADDUI R3, R3, #-8
       DADDUI R4, R4, #-8
       DADDUI R5, R5, #-8
       BNE R2, R1, Loop
       SD F5, 8(R5)
```

Superscalar Processors: For the following question, consider a superscalar processor which supports two in-order pipelines with bypassing (one instruction in each pipeline stage in any cycle). One pipeline handles only L.D, S.D, ADDUI, and BNE instructions and the second pipeline is designed to handle only FP ADD and FP MULT. Assume the following delays between dependent instructions:

(a) Load feeding any instruction: 1 stall cycle
(b) FP MUL/ADD feeding store: 4 stall cycles
(c) FP MUL/ADD feeding any other instruction: 1 stall cycles
(d) Integer ADD feeding a branch: 1 stall cycle
(e) A conditional branch has 1 delay slot (an instruction is fetched in the cycle after the branch without knowing the outcome of the branch and is executed to completion)

3. **Software Optimization:** Below is the source code and default assembly code for a loop.

Source code

```
for (i=1000;i>0;i--) {
x[i] = s * y[i] + z;
}
```

Assembly code

```
Loop: L.D F1, 0(R1)
F.MUL F3, F1, F2
F.ADD F5, F3, F4
S.D F5, 0(R2)
DADDUI R1, R1, #-8
DADDUI R2, R2, #-8
BNE R1, R3, Loop
NOP
```

i. Show an optimized schedule for this loop without resorting to unrolling. **(10 points)**

Answer:

Integer Pipeline	Floating Point Pipeline
Loop: LD F1, 0(R1)	NOP
DADDUI R1, R1, #-8	NOP
NOP	F.MUL F3,F1,F2
DADDUI R2, R2, #-8	NOP
NOP	F.ADD F5, F3, F4
	Stall
	Stall
	Stall
BNE R1,R3, Loop	NOP
SD F5, 8(R2)	NOP

ii. Optimize the schedule for this loop using loop unrolling. How many unrolls does it take to avoid stall cycles?**(15 points)**

Integer Pipeline	Floating Point Pipeline
Loop: LD F1, 0(R1)	NOP
LD F6, -8(R1)	NOP
LD F11, -16(R1)	FMUL F3, F1, F2
	FMUL F8, F6, F2
	FMUL F13, F11, F2
	FADD F5, F3, F4
	FADD F10, F8, F4
	FADD F15, F13, F4
DADDUI R1, R1, #-24	NOP
DADDUI R2, R2, #-24	NOP
SD F5, 24(R2)	NOP
SD F10, 16(R2)	NOP
BNE R1, R3, Loop	NOP
SD F15, 8(R2)	NOP

It takes 3 unrolls to avoid stall cycles – or cycles in which no pipeline is stalled