CS 356 Processor Organization

Redekopp

Name: _____

Score: _____

1.) Given a loop such as that shown below with the given assembly translation.

```
void f1(int* A, int* B, int N)
for( ; N != 0; A--,B--,N--){
  temp = *A;
  *A = temp + *B + 9;
  *B = temp;
}
```

```
L1:  ld (%rdi),%eax   ; load temp=*A
     ld (%rsi),%ebx   ; load *B
     add %eax,%ebx    ; add temp+*B
     add $9,%ebx      ; add 9
     st  %ebx,(%rdi)  ; store *A
     st  %eax,(%rsi)  ; store *B
     add $-4,%rdi     ; dec. A ptr.
     add $-4,%rsi     ; dec. B ptr.
     add $-1,%rdx
     jne $0,%rdx,L1 ; loop
```

a.) (15 pts.) Schedule the code for our 2-way VLIW (static multiple issue) superscalar discussed in lecture (1 ALU/Branch slot and 1 load/store slot). You may move code up or down and modify it if it helps scheduling. You cannot use register renaming or loop unrolling in this part. Once finished calculate the IPC.

ALU / Branch	Load / Store

IPC = 10 instructions / _____ clocks = _____ IPC

b.) (25 pts.) Now unroll the loop for an additional iteration (2 iterations per body) and use register renaming where appropriate. When performing register renaming, use register numbers %r8d, %r9d, %r10d, … in that order so that everyone's answer will hopefully be more uniform. Schedule the code for our 2-way VLIW (static multiple issue) superscalar discussed in lecture (1 ALU/Branch slot and 1 load/store slot). You may modify the execution order (up or down) and change ld/st offsets if it helps scheduling. Assume full forwarding. Once finished calculate the IPC.

```
L1:  ld (%rdi),%eax   ; load temp=*A
     ld (%rsi),%ebx   ; load *B
     add %eax,%ebx    ; add temp+*B
     add $9,%ebx      ; add 9
     st  %ebx,(%rdi)  ; store *A
     st  %eax,(%rsi)  ; store *B
     ld  _____________
     ld  _____________
     add _____________
     add _____________
     st  _____________
     st  _____________
     add $-8,%rdi     ; dec. A ptr.
     add $-8,%rsi     ; dec. B ptr.
     add $-2,%rdx
     jne $0,%rdx,L1 ; loop
```

ALU / Branch	Load / Store

IPC = 16 instructions / _____ clocks = _____ IPC

2.) (12 pts.) Given the code below, perform explicit register renaming to solve all WAW, WAR hazards present in the original code. When performing register renaming, use register numbers %r8, %r9, %r10… in that order so that everyone's answer will hopefully be more uniform.

`ld  0(%rdi), %rax`
`add  %rbx,%rax`
`sub  %rdx,%rcx`
`ld  0(%rsi),%rbx`
`sub  %rbx,%rdx`
`mov  %rsp,%rsi`
`add  %rcx,%rsi`
	`ld  0(_____), _____`
	`add  _____, _____`
	`sub  _____, _____`
	`ld  0(_____), _____`
	`sub  _____, _____`
	`mov  _____, _____`
	`add  _____, _____`

3.) (5 pts.) Given the code below, assume the first 'ld' instruction stalls due to a cache miss. Assuming an out-of-order, **dynamically** scheduled processor (that performs automatic register renaming), which instructions would be allowed to execute (i.e. are independent) and which instructions would need to stall due to the 'ld'.

Code	**Circle the correct answer**
`ld  0(%rdi), %rax`	CACHE MISS
`add  %rbx,%rax`	Stall / Execute
`sub  %rax,%rcx`	Stall / Execute
`ld  0(%rsi),%rbx`	Stall / Execute
`sub  %rbx,%rdx`	Stall / Execute
`add  %rcx,%rsi`	Stall / Execute