COMP 222
Sample Final Exam

Problem 1 (15 points)

Assuming the following address and register contents, determine at least fifteen possible ways to put the value 9 into register R1, assuming instructions of the form:
LOAD ___, R1. For each way, describe what type of addressing mode you are using. Use only the addressing modes presented in class (Immediate, Direct, Indirect, Register, Register Indirect, Displacement). (Note: additional ways beyond fifteen will count +1 if correct and -1 if incorrect. Add more rows to the table if needed).

Addr 1	Addr 2	Addr 3	Addr 4	Addr 5	Reg R1	Reg R2	Reg R3	Reg R4	Reg R5
9	4	1	9	1	1	9	9	4	0

Source operand	Addressing Mode
#9	Immediate
R2	Register
R3	Register
1	Direct
4	Direct
(3)	Indirect
(2)	Indirect
(5)	Indirect
(R1)	Register indirect
(R4)	Register indirect
(R1+0)	Displacement
(R1+3)	Displacement
(R4+0)	Displacement
(R5+1)	Displacement
(R5+4)	Displacement

Problem 2 (20 points)

Suppose that a program is run on different architectures. Assume that there are no branches, but read-after write dependencies between adjacent instructions ***i*** and ***i+1*** occur on average 30% of the instructions, and that read-after-write dependencies between instructions ***i*** and ***i+2*** occur on average 25% of the instructions (assuming no overlapping of dependencies)..

a) Determine the expected total number of cycles for the above program run on a 6-stage pipeline, consisting of *n* instructions. What is the speedup compared to running the program on a 6-stage non-pipelined architecture, as $n \to \infty$

Extra cycles between ***i*** and ***i+1*** : 2 (delay between WO cycle of ***i*** and FO cycle of ***i+1***)
Extra cycles between ***i*** and ***i+2*** : 1 (delay between WO cycle of ***i*** and FO cycle of ***i+2***)
T = (k+n-1)+(sum of extra delays) = (n+5)+(30%)(n)(2)+(25%)(n)(1)=1.85n+5 cycles

Speedup: lim (n→∞) (no pipeline / pipeline) = lim (n→∞) ((6n / 1.85n+5)) = 3.24

b) Determine the expected total number of cycles for the above program run on a 6-stage superpipeline, consisting of the same *n* instructions. What is the speedup compared to running the program on a 6-stage non-pipelined architecture, as $n \to \infty$

Extra cycles between ***i*** and ***i+1*** : 2.5 (delay between WO cycle of ***i*** and FO cycle of ***i+1***)
Extra cycles between ***i*** and ***i+2*** : 2 (delay between WO cycle of ***i*** and FO cycle of ***i+2***)
T = (k+(n-1)/2)+(sum of extra delays) = (0.5n+5.5)+(30%)(n)(2.5)+(25%)(n)(2)=1.75n+5 cycles

Speedup: lim (n→∞) (no pipeline / pipeline) = lim (n→∞) ((6n / 1.75n+5)) = 3.43

c) Determine the expected total number of cycles for the above program run on a 6-stage superscalar, consisting of the same *n* instructions. What is the speedup compared to running the program on a 6-stage non-pipelined architecture, as $n \to \infty$

Extra cycles between ***i*** and ***i+1*** : 2.5 (delay between WO cycle of ***i*** and FO cycle of ***i+1***) (actually either 3 or 2, depending on the grouping, so 2.5 on average)
Extra cycles between ***i*** and ***i+2*** : 2 (delay between WO cycle of ***i*** and FO cycle of ***i+2***) (independent of grouping of instructions)
T = (k+$\lceil$ n-1)/2 $\rceil$)+(sum of extra delays) ≈
(0.5n+5.5)+(30%)(n)(2.5)+(25%)(n)(2)=1.75n+5 cycles

Speedup: lim (n→∞) (no pipeline / pipeline) = lim (n→∞) ((6n / 1.75n+5)) = 3.43

Problem 3 (20 points)

Suppose that a program is tested for the performance of *n* instructions on a 6-stage pipeline architecture, where the probability a conditional branch is predicted taken or not taken depending on the direction of the branch. All backward branches (which decrease the program counter PC) are predicted taken. All forward branches (which increase the program counter PC) are predicted not taken. The probability of a backward conditional branch is 9% of all instructions, the probability of a forward conditional branch is 6% of all instructions, and the probability of an unconditional branch is 4% of all instructions.

a) Determine the expected total number of cycles to process all *n* instructions, assuming that prediction is correct 70% of the time, and that: (i) an unconditional branch stalls the pipeline for 2 extra cycles; (ii) a correct prediction of "never taken" stalls the pipeline for 0 extra cycles; (iii) a correct prediction of "always taken" stalls the pipeline for 2 extra cycles; and (iv) an incorrect prediction stalls the pipeline for 4 extra cycles, where an extra cycle is defined as a delayed cycle that is not a normal pipeline cycle. Determine the speedup compared to a non-pipelined architecture. as $n \rightarrow \infty$.

T=(k+n-1)+(4%)(n)(2) [Unconditional branch]
+ (9%)(n)(2)(70%) [Correctly predicted backward branch]
+(6%)(n)(0)(70%) [Correctly predicted forward branch]
+(15%)(n)(4)(30%) [Incorrectly predicted branch]
=n+5+0.08n+0.126n+0.18n=1.386n+5 cycles

Speedup: lim (n→∞) (no pipeline / pipeline) = lim (n→∞) ((6n / 1.386n+5)) = 4.33

b) Suppose that we have a goal to achieve a speedup of at least 4.5 compared to a non-pipelined architecture. as $n \rightarrow \infty$. What should be the minimum percent of correct prediction to achieve this goal.

Let p be the percent of correct prediction we want to find.
T=(k+n-1)+(4%)(n)(2) [Unconditional branch]
+ (9%)(n)(2)(p) [Correctly predicted backward branch]
+(6%)(n)(0)(p) [Correctly predicted forward branch]
+(15%)(n)(4)(1-p) [Incorrectly predicted branch]
=n+5+0.08n+0.18np+0.6n(1-p)=n(1.68-0.42p)+5 cycles
Speedup: lim (n→∞) (no pipeline / pipeline) = lim (n→∞) (6n / n(1.68-0.42p)) = 4.5
6 / (1.68-0.42p) = 4.5 → 1.68-0.42p = 1.33 → 0.42p = 0.35 → p = 83.33%

Problem 4 (20 points)

Consider the following register instructions which are processed on a 6-stage pipeline architecture.

1) r3=r2+r4
2) r4=r3+r1
3) r5=r4+r2
4) r2=r1+r3
5) r1=r0+r3
6) r3=r0+r0

a) Determine the optimal number of cycles if there were no dependencies at all (i.e. perfect pipeline):

T=n+k-1 = 6+6-1=11 cycles

b) Determine the actual number of total cycles assuming no reordering of instructions.

Delay from 1) to 2): 2
Delay from 2) to 3): 2

T=n+k-1+(sum of extra delays)=6+6-1+(2+2)=15 cycles

c) Determine the best achievable number of cycles assuming instruction reordering with possible **register renaming**, showing the new set of instructions with the **original numbering.**

1) r3=r2+r4
6) t3=r0+r0 [Renamed r3 to t3]
2) r4=r3+r1
4) t2=r1+r3 [Renamed r2 to t2]
5) r1=r0+r3
3) r5=r4+r2

Delay from 1) to 2): 1
T=n+k-1+(sum of extra delays)=6+6-1+1=12 cycles