Computer Architecture
PhD Qualifier Exam Examples

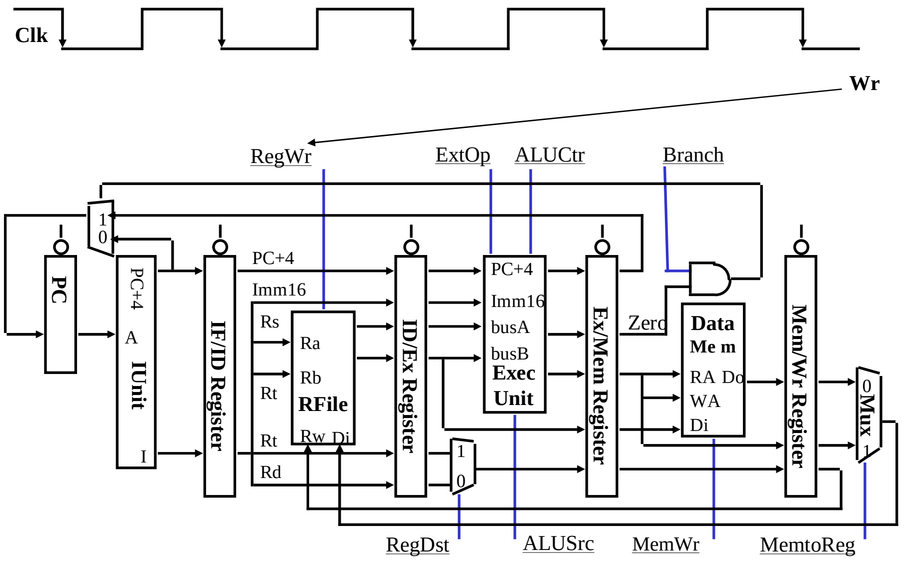
Problem (Pipeling Datapath & Control):

Figure 1 illustrates a 5-stage MIPS pipeline datapath. One *Load*, *R-type (ALU)*, *Store*, *Beq* instruction enters the datapath in Cycle 1, 2, 3, and 4, respectively. Suppose each instruction takes all 5 pipelining stages, i.e., Instruction fetch (IF), Instruction decoding / register fetch (ID/RF), ALU execution (Exec), memory access (Mem), and Register write-back (Wr). Assume there is neither control hazard nor data hazard.

a) Please set the control values in the following table in the end of Cycle 4, 5, and 6, respectively.

	RegDst	RegWr	ExtOp	ALUCtr	MemWr	MemtoReg	Branch
Cycle 4							
Cycle 5							
Cycle 6							

Figure 1: A Pipelined Datapath



Consider the following piece of codes on the pipelining machine.

```
Lw  $1, 100 ($2)
Add $3, $5, $1
Sub  $4, $5, $6
Add $7, $1, $4
Sub  $8, $9, $7
```

b) Assuming there is neither forwarding logics inside the CPU nor compiler help, please draw a diagram with the five pipelining stages to show all of the data hazards. And, how many *no*p instructions need to be inserted?

c) Assuming there are forwarding logics inside the CPU, draw a diagram to show how many data hazards can be resolved, and how many are still there.

d) With the forwarding logics inside the CPU, is it possible to reorder (schedule) the instruction sequence so as to remove data hazards to zero? If yes, show your instruction order?

Problem (Memory Hierarchy):

a) A cache system takes advantage of locality to reduce the average access time to the main memory. The principle of Locality states the programs access a small portion of address space at any instant of time.

What are the two types of locality used in building a cache system? And, use the following program code to give a concrete example for each type of locality.

```
i = 0;
Sum = 0;
While (i < 100) {
        Sum += A[i];                    // A and B are two arrays, C is a variable
        B[i] += C;
        i ++;
}
```

b) Given a series of references in memory word addresses: 1, 4, 7, 5, 12, 9, 11, 20, 9, 13, 4, 5. A cache is empty initially.

Show the hits and misses, and the final cache content for a directed mapped cache with a total of 8 one-word blocks, and for a directed mapped cache with a total of 4 two-word blocks, respectively.

c) Assuming a 32-bit memory address and using block size of 2^m bytes, calculate the **total bits** required for a N-way set associative cache with 2^M bytes of data (N = 2^k); M, N, m, k all integers.

d) Assuming an instruction cache miss rate for *gcc* of 2% and a data cache miss rate for *gcc* of 4%. If a machine (M1) has a CPI of 2 if without any memory stalls and the miss penalty is 40 cycles for all kinds of misses, determine how much faster a machine (M2) that runs with a perfect cache that never missed. The instruction mix of *gcc* is given in the following table.

Instruction class	***Frequency***
R-type	50%
Load	20%
Store	16%
Branch	14%

If we double clock rate of M1 so as to get a machine M3, assuming the absolute time (miss penalty) to handle a cache miss does not change, how much faster M3 than M1?

Pick any two of the three questions below:

1. For the CFG with the following production rules (Show all steps in each part)

$S \to aC \mid ABaD \mid BD$; $A \to aAAb \mid a \mid b \mid aC$; $B \to abB \mid aD$;
$C \to aCb \mid b \mid aB$; $D \to aaD \mid abB$

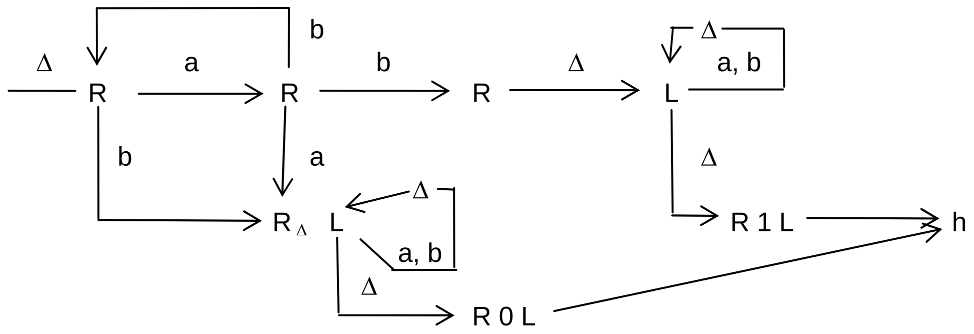
(a) Put the CFG in its simplest form (for human readability) by eliminating all useless variables and production rules.
(b) What set of strings can S derive under closure? Use precise notation.

2. For the language $L = \{ x \mid x \in \{a, b\}^* \mid N_a(x) < N_b(x) \}$, find an accepting PDA.

(a) State the logic you use in constructing the PDA.
(b) Draw the transition diagram.
(c) Include an error state with all error transitions.

3. For the following TM

a) What language L is accepted by the TM? Give your answer in set notation. It must be in simplest form for full credit. Note: Δ is a blank symbol.
b) How would you best characterize exactly what the TM does?



Operating Systems sample questions

Deadlock:

A) Consider a system with 19 disk drives. Each user will never need more than four at a time. Let N be the number of processes. For what values of N is the system deadlock free if we always grant requests for available drives? Why? Does the value of N change if we use a deadlock avoidance algorithm such as, for example, the banker's algorithm to decide whether to grant a request? Why?

B) Terms usually bandied about in the discussion of deadlock include:

Avoidance
Circular Wait
Detection and Recovery
Exclusive Access
Hold and Wait
No Preemption
One Shot Allocation
Prevention
Resource Ordering

Try to group them into categories. Define and explain why you have chosen each of the categories you select.

Scheduling:

A) Suppose your computer uses a multi-level feedback queue to schedule processes. Assume that the queue has the following priority and timeout values and that newly ready-to-run processes enter at priority level 1:

Priority	Time Slice
1	10
2	20
3	40
4	80

Suppose a process repeatedly does 12 ms of computing followed by 10 msec of I/O followed by 35 msec of computing followed by 70 msec of I/O. Describe the path it takes through the scheduler queues.

B) Suppose this system implements a Translation Lookaside Buffer (TLB) and paged page tables in hardware. Suppose that the TLB misses on I % of all virtual memory references, that access to the TLB itself is essentially 0 ns, and that accesses to physical memory take 200 ns. What is the average time required to return data for a virtual memory reference? State any assumptions you make.

PhD Qualifier – Algorithms

Submission requirements:

Answer exactly 2 of the following 3 questions and work them to the best of your ability. If you are not sure about something, explain the parts of it you *do* understand as clearly as possible and move on. Partial credit is possible, but remember that you must demonstrate a correct understanding of the underlying concepts. Be clear about any assumptions that you make.

Questions:

1. *Recurrence Relations:*

   (a) Determine the asymptotic efficiency class of each of the following recurrence relations. That is, determine $\Theta(\cdot)$ for each of the functions T_1, T_2, etc. For each, you can assume that $T(1) = c_1$ and $T(0) = c_0$ where $c_1 \geq c_0 > 0$.

   i. $T_1(n) = 2T_1(n-1)$
   ii. $T_2(n) = 2T_2(n-2)$
   iii. $T_3(n) = 5T_3(n-2)$
   iv. $T_4(n) = T_4(n-1) + T_4(n-2)$

   (b) Rank the above recurrences in order of complexity.

2. *Greedy Algorithms:* Consider the following *change-making problem*. You have an amount (of money) $n > 0$ and coins of the denominations used in the United States: $\{1, 5, 10, 25\}$. Define a *greedy change-making algorithm* as one which uses as many of the largest coin as possible first, then as many of the 2nd-largest coin as possible, and so on – until the proper change is made. For example, to make change for $n = 41$, the greedy algorithm will first select a 25-cent coin (leaving 16 cents remaining), then a 10-cent coin (leaving 6 cents), then a 5-cent coin, then a 1-cent coin.

   (a) Prove that for U.S. coins, the greedy algorithm above always use the fewest-possible coins to make change for any n. That is, this greedy algorithm is optimal.

   (b) Is the above greedy algorithm optimal in general for the change-making problem? That is, for an arbitrary set of denominations $d_1 < d_2 < \cdots < d_m$, will the greedy algorithm always select the smallest number of coins to make change for any n? Prove or disprove your statement.

(continued on next page)

3. *P, NP, and NP Complete Problems:* Answer all of the following parts to receive credit for this question:

   (a) Is the following statement True or False? *Let* `PROB` *be an NP-Complete problem. There is a deterministic algorithm with polynomial best-case running time that solves* `PROB` *exactly.* Justify your answer.

   (b) Suppose that problem `PROB1` can be reduced in polynomial time to problem `PROB2` and that the best-known deterministic algorithm for solving `PROB1` has $\Theta(2^n)$ running time. Given this information, what can you say about the complexity classes of `PROB1` and `PROB2`? Is it possible that there exists a polynomial-time algorithm for `PROB2`?

   (c) One of the great open questions of mathematics is to determine whether $P = NP$ or $P \neq NP$. There are several possible proof approaches that could resolve this question. Outline 3 hypothetical approaches. That is, what would a proof look like that $P = NP$? What would a proof look like that $P \neq NP$?