Qualifying exam (Computer Architecture)

Sample problems:

Problem 1:

a) Define Amdahl's law**.**

b) As a computer architect and with respect to the Amdahl's law, what are your observations in improving the performance of a system (computer).

c) How do you use your observation(s) in (b) to design a fast multiplier-unit.

d) How are your observation(s) in (b) used in the design of the so-called column-compression multiplication method.

e) Apply column-compression method to perform the following operation:
1110011 * 0111011

Problem 2:

Booth algorithm is a technique that allows multiplication of two 2^s complement numbers:

a) True or false; On the average Booth algorithm is faster than add-and-shift algorithm (justify your answer)?

b) Booth algorithm can be extended by checking three bits of multiplicand in one loop iteration. Compare and contrast traditional Booth algorithm with **extended Booth algorithm**.

c) Extended Booth algorithm can be further modified by checking group of more than three bits in each iterations (say 4, 5, …). However, in practice rarely, Booth algorithm based on grouping of more than three bits has been implemented. Why (clear explanation)?

d) Apply **extended Booth algorithm** (group of 3 bits) to perform the following multiplication:
   a. Multiplier 1000111
   b. Multiplicand 1111000

Problem 3:

SRT method is a technique that improves division of the floating point numbers:

a) Clearly explain it and make sure to indicate how it improves the performance,

b) Apply SRT technique to perform the following operation (show step by step operation, make sure that each step is well documented):
$(.1110010000*2^5) / (.10111*2^2)$

Problem 4:

Multiplication of two n-bit numbers (n a power of 2) can be carried out as 4 multiplications of n/2-bit numbers and two additions:

a) Clearly prove this.
b) Discuss about the advantages of this approach to multiplication.
c) Assuming 2X2 multiplier units, extend the scope of your proof in (a) to multiply two 8X8 numbers (show your work as clearly as possible).
d) Based on your answer in part (c) multiply the following numbers using 2X2 multiplier units (note numbers are in 2^s complement notation:
    a. Multiplier 10001110
    b. Multiplicand 10110001

Problem 5:

a) Derive an equation to calculate the CPU time in terms of clock period, # of clock cycles, and # of instructions. (Justify your approach).
b) With respect to the equation for the CPU time, compare and contrast RISC and CISC design philosophies against each other.
c) With respect to the equation for the CPU time, define design principal(s) behind:
    a. Super-scalar architecture,
    b. Very Long Instruction Word architecture, and
    c. Super Pipeline architecture.
d) Compare and contrast super-scalar architecture against VLIW architecture.
e) Within the scope of super-scalar and VLIW architectures, name and discuss (briefly) ***distinct*** issues that do not allow ideal performance.
f) Within the scope of super-scalar and VLIW architectures, name and discuss (briefly) some solutions that remedy the effect of issues examined in (e)

Problem 6:

a) Define the term “execution time”.
b) Define the term “user CPU time”.
c) Relate user CPU time to the execution time.
d) Develop an equation to calculate user CPU time showing the role of ALU, control unit, and effect of the memory latency.
e) Relate RISC design philosophy with your answer in part (d).
f) It has been argued that RISC philosophy allows efficient **use** and **management** of instruction pipeline, justify this as clearly as possible.

Problem 7:

a) As a performance metric define MIPS
b) It has been discussed that MIPS is not an accurate measure for comparing performance among computers justify this. Note, I need a discussion that is straight-forward and clear.

Problem 8:

a) Define the term “speed up”,
b) Define the term “scale up”,
c) For a parallel system composed of *p* processors, define the term “Efficiency”,
d) With respect to the equation defined in part “c”, explain your observation (s),
e) For a parallel system of *p* processors, develop an algorithm to add up *p* numbers:
    a. Define your algorithm,
    b. Calculate the speed up for your algorithm, and
    c. Calculate the efficiency of your algorithm.

Problem 9:

a) Define term “interleave memory”,
b) Define high-order interleaving,
c) Define low-order interleaving,
d) Compare and contrast high-order interleaving with low-order interleaving,
e) Two issues affect performance of an interleave memory:
    a. What are they,
    b. Show (proof) how do they affect the effectiveness of the interleave memory
f) With respect to the part (e), discuss about solutions (one for each case).

Problem 10:

c) Define cache coherence,
d) Within the scope of a uni-processor organization, does the cache coherence issue exist (justify your answer)?
e) If your answer to part (b) is yes, discuss how a write policy (write-back and write-through) will affect cache coherency issue.

Problem 11:

Within the scope of a multiprocessor system, partitioning, scheduling, and load balancing are common practices that allows performance, say the throughput and hardware utilization, improvement. In partitioning, a task is partitioned into subtasks and subtasks are scheduled for execution on different processors.

a) Within the scope of partitioning, scheduling, and load balancing **clearly** name and explain issues of concern (at least two issues).
b) Clearly examine potential solutions for the issues addressed in part (a).

Problem 12:

Loop fusion allows two or more loops that are executed the same number of times and that use the same indices to be combined into one loop:

a) Within the scope of a RISC processor, why does it (**Loop fusion**) improve performance (detail explanation)?
b) Within the scope of a vector processor, why does it (**Loop fusion**) improve performance (beyond what has been discussed in **Part a**) (detail explanation)?
c) Within the scope of a superscalar processor, why does it (**Loop fusion**) improve performance (beyond what has been discussed in **Part a**) (detail explanation)?

Problem 13:

g) Define the term “Super-scalar architecture”,
h) Define the term “Very Long Instruction Word architecture”,
i) Compare and contrast super-scalar architecture against VLIW architecture,
j) Given the following assembly code: Exploit the maximum degree of parallelism among the 16 instructions, assuming no resource conflicts and availability of multiple functional units. For simplicity, no pipelining is assumed:
   a. Draw the program graph to show the follow relationships among the 16 instruction nodes.
   b. Show the representation of this program in a VLIW, where each instruction word could have up to two memory reference operations, one add/subtract operation, and one Mult operation (I need detail and precise presentation). All instructions take one machine cycle to execute, and ignore all other overhead.
   c. Same as “b”, however, Load and Store operations take three machine cycles to execute, Add and Subtract operations take one machine cycle to execute, and Mult. Operations take two machine cycles to execute (ignore all other overheads).

1)	LOAD	R_1, A
2)	LOAD	R_2, B
3)	MUL	R_3, R_1, R_2
4)	LOAD	R_4, D
5)	MUL	R_5, R_1, R_4
6)	ADD	R_6, R_3, R_5
7)	STORE	X, R_6
8)	LOAD	R_7, C
9)	MUL	R_8, R_7, R_4
10)	LOAD	R_9, E
11)	Add	R_{10}, R_8, R_9
12)	STORE	Y, R_{10}
13)	ADD	R_{11}, R_6, R_{10}
14)	STORE	U, R_{11}
15)	SUB	R_{12}, R_6, R_{10}
16)	STORE	V, R_{12}