PRACTICE PROBLEMS

COMPUTER ORGANIZATION AND ARCHITECTURE
DESIGNING FOR PERFORMANCE
TENTH EDITION

WILLIAM STALLINGS

TABLE OF CONTENTS

CHAPTER 1 BASIC CONCEPTS AND COMPUTER EVOLUTION

1.1 Let **A** = A(1), A(2), . . ., A(1000) and **B** = B(1), B(2), . . . B(1000) be two vectors (one-dimensional arrays) comprising 1000 numbers each that are to be added to form an array C such that C(I) = A(I) + B(I) for I = 1, 2, . . ., 1000. Using the IAS instruction set, write a program for this problem. Ignore the fact that the IAS was designed to have only 1000 words of storage.

CHAPTER 2 PERFORMANCE ISSUES

2.1 Consider a machine running at 2GHz. The program instructions in this machine are categorized in the following four classes. A programmer uses two different optimizations of a compiler generating two different machine codes. Compute the execution time and MIPS for each code and decide which optimization results in faster program.

Class	I	II	III	IV
CPI	1	2	3	4

Class	I	II	III	IV
Optimization 1	10	3	4	3
Optimization 2	20	3	2	1

2.2 Suppose that we are considering an enhancement to the processor of a server system used for web serving. The new CPU is 10 times faster on computation in the web serving application than the original processor. Assuming that the CPU is idle for 20% of time, does computation for 30% of time and is waiting for I/O for 50% time, what is the overall speed up in this case?

2.3 Consider the following table, showing the relative execution times and absolute mflops execution rates for a benchmark suite of five programs. The relative execution time of program *i* is defined as the ratio of its execution time t_i to the sum T of all execution times within the suite.

Program	t_i/T	MFLOPS
P1	0.10	1000
P2	0.55	110
P3	0.20	500
P4	0.10	200
P5	0.05	75

a. Consider these 5 programs in totality and write the best possible way to combine them to derive a single measure of merit.
b. Using the answer for (a), derive a mean execution rate for the benchmark suite described in table 1.
c. If the execution rate of all programs except P2 can be increased arbitrarily by increasing the speed of the floating point units, what is

the upper bound on the execution rate of the benchmark suite as a whole?

2.4 Assume we make an enhancement to a computer that improves some mode of execution by a factor of 15. This new fast mode is used 40% of the time, measured as a percentage of the execution time when the fast mode is in use. What is the overall speedup we can achieve?

2.5 Imagine that you are the lead architect on a team assigned to design a new multiprocessor computer. The goal for this new product is for it to be a cost-performance leader within the server computing market. The marketing team has already specified that the total cost of the machine must fall somewhere within the range of the $50,000 - $100,000 that is typical of high-end server machines, and that furthermore it ought to be as close as possible to the low end of this range, so as to best attract the value-seeking customer. You expect that the purchasers in the market for this particular product will not care very much about power consumption, as long as it is less than about 50 kW.
You are trying to choose between two different types of processors to base your system design on. Chip A costs $2,750, performs at a throughput level of 80 work units (database transactions) per second in a benchmark application, and consumes 100W of power. Chip B costs $1,600, completes 50 work units per second, and consumes 50W of power. (For this problem, neglect the cost and power consumed by other components.)
In the following, use these variables:

$C_{sys,min}$ = The minimum cost of the system
$C_{sys,max}$ = The maximum cost of the system
$P_{sys,max}$ = The maximum power consumption of the system.
C_X = Cost of a chip of type X (where X = A or B)

(And similarly with *P* (power) and *T* (throughput) of chips A and B)

n_{chips} = Number of chips of the selected type in the system.

a. Identify the design constraints that our system design must satisfy. Also, assuming that power will not end up being the limiting factor in the design, identify the quantities that we should be trying to maximize and/or minimize in this design scenario.

b. Now, formulate analytical expressions, in terms of the variables given above, for:
(i) the constraints that our design must satisfy
(ii) the quantity that we should be maximizing in our design (figure of merit)

(iii) the quantity that we should be minimizing in our design (figure of demerit)

c. Answer the following questions and show your work.
(i) Which type of chip (A or B) does a better job of maximizing the figure of merit?
(ii) How many copies of this chip should we include in our system design? That is, what is the optimal value of n_{chips}, within the design constraints?
(iii) Does the chip that you chose in part (i) also minimize the figure of demerit, within the design constraints?

2.6 Research Gustafson's law on the Internet. Explain the applicability and the restrictions involved in using Amdahl's law and Gustafson's law to estimate the speedup performance of an n-processor system compared with that of a single-processor system. Ignore all communication overheads.

2.7 Computer A executes the MIPS ISA and has a 2.5Ghz clock frequency. Computer B executes the x86 and has a 3Ghz clock frequency. On average, programs execute 1.5 times as many MIPS instructions than x86 instructions.

a. For Program P1, Computer A has a CPI of 2 and Computer B has a CPI of 3. Which computer is faster for P1? What is the speedup?

b. For Program P2, Computer A has a CPI of 1 and Computer B has a CPI of 2. Which computer is faster for P2? What is the speedup?

2.8 A program has 10% divide instructions. All non-divide instructions take one cycle. All divide instructions take 50 cycles.

a. What is the CPI of this program on this processor?

b. What percent of time is spent just doing divides?

c. What would the speedup be if we sped up divide by 2x?

d. What would the speedup be if we sped up divide by 5x?

e. What would the speedup be if we sped up divide by 10x?

f. What would the speedup be if we sped up divide by 50x?

g. What would the speedup be if divide instructions were infinitely fast (zero cycles)?

2.9 A program runs in 200ms and 60% of instructions are divide operations. We want to design a new divider module in order to have the programmer run 2.5 times faster. How much speed up is needed in new divider?

2.10 We have an accounting application with the execution time of 100 seconds. The program has two main functions "Sort" and "Computation". Half of instructions are coded for implementing Sort function and 15% for Computation. In Sort function 40% of

instructions are swap (data movement), 25% subtract and 20% add. In Computation function instructions are 15% floating point operations and 50% integer multiplication. We have two proposals to enhance the application performance.

A) We can optimize the data movement path so that swap operation performs twice as fast.
B) The floating-point operations can be redesigned in a way to get a speed up of 4 times for them.

Considering the following table illustrating the CPI for each operation, compute the speed up for each proposal and compare them.

Operation	add	subtract	swap	Integer multiplication	Floating point	others
CPI	1	1	2	3	10	1.5

CHAPTER 3 COMPUTER FUNCTION AND INTERCONNECTION

3.1 Compare buses, crossbar switches, and multistage networks for building a multiprocessor system. Assume a word length of *w* bits and 2 × 2 switches are used in building the multiprocessor system with *n* processors and *m* shared-memory modules. Assume a word length of *w* bits and that 2 × 2 switches are used in building the multistage networks. Show your answer in a table with the following format:

Network Characteristics	Bus System	Multistage Network	Crossbar Switch
Minimum latency for unit data transfer			
Bandwidth per processor			
Wiring complexity			
Switching complexity			
Connectivity and routing capability			
Remarks			

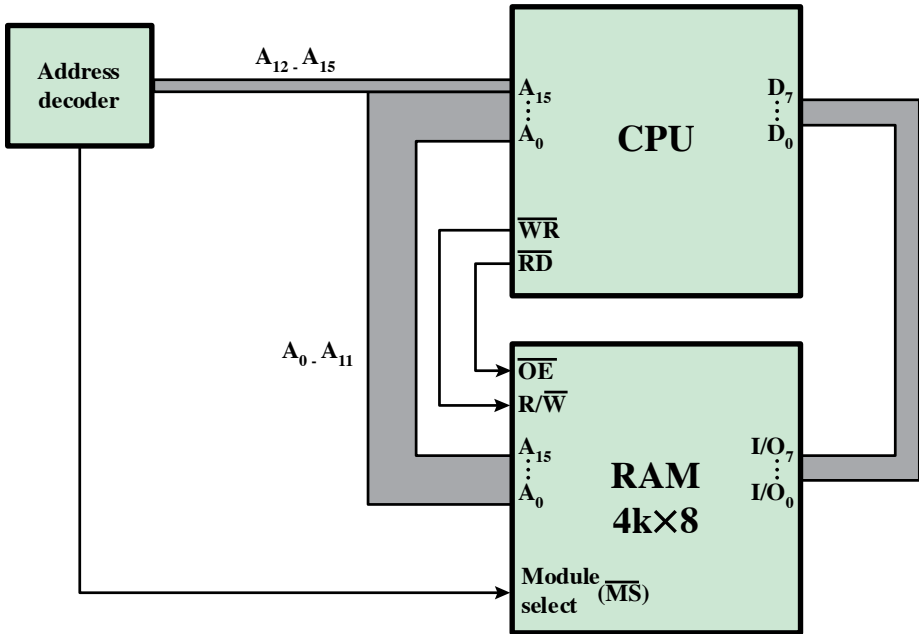
3.2 **a.** Consider a main memory system that consists of a number of memory modules attached to the system bus, which is one word wide. When a write request is made, the bus is occupied for 100 nanoseconds (ns) by the data, address, and control signals. During the same 100 ns, and for 500 ns thereafter, the addressed memory module executes one cycle accepting and storing the data. The (internal) operation of different memory modules may overlap in time, but only one request can be on the bus at any time. What is the maximum number of stores that can be initiated in one second?

b. Sketch a graph of the maximum write rate (words per second) as a function of the module cycle time, assuming eight memory modules and a fixed bus busy time of 100 ns.

3.3 Suppose a bus protocol requires 10 ns for devices to make requests, 15 ns for arbitration, and 25 ns to complete each operation. How many operations can be completed per second?

3.4 A given processor has eight interrupt lines (numbered 0 - 7), and a policy that low-numbered interrupts have priority over higher-numbered ones. The processor starts with no interrupts pending, and the following sequence of interrupts occurs: 4, 7, 1, 3, 0, 5, 6, 4, 2, 1. Assume that handling any interrupt takes enough time that two more interrupts arrive while the first interrupt is being handled, until all of the interrupts have arrived, and that interrupts cannot interrupt each other. What order are the interrupts handled in?

3.5 Consider the processor and RAM portion of a system depicted in the following figure;



a. The RAM contains how many words at what bit length?

b. For the RAM to be written into, address inputs A_0 through A_{11} must be activated, then $\overline{MS}$ must be enabled. Should that be with a HIGH or a LOW? At the same time the R/$\overline{W}$ held at a constant level; should that be a HIGH or a LOW? Should the $\overline{OE}$ be driven HIGH or LOW? Finally, are I/O_0 through I/O_7 inputs or outputs for the operation?

c. What lines constitute the control bus in the figure?

CHAPTER 4 CACHE MEMORY

4.1 A certain memory system has a single level of cache, connected to a main memory. The time to access memory at an address that is present in the cache is t_h, whereas the time for the same access when the address is not present in the cache is t_m. Derive an equation for E, the effectiveness of the cache, defined as the ratio t_h/t_a, where t_a is the mean access time for all hits and all misses over the measurement interval.

4.2 Cache-Memory Mapping: Consider a memory of 32 blocks (labeled 0 through 31) and a cache of 8 blocks (labeled 0 through 7). In the questions below, list only correct results.

a. Under direct mapping, which blocks of memory contend for block 2 of the cache?

b. Under 4-way set associativity, to which blocks of cache may element 31 of memory go?

c. In the following sequence of memory block references from the CPU, beginning from an empty cache, on which reference must the cache layout for a direct-mapped and a 4-way set associative cache first differ?

order of reference	1	2	3	4	5	6	7	8
block referenced	0	15	18	5	1	13	15	26

d. Suppose a direct-mapped cache is equipped with a single-block victim cache. Out of the following sequence of memory block references from the CPU (R denotes read (or load) and W denotes write (or store)), on which references is the block retrieved from the victim cache? If no block is ever thus retrieved, so state. Show your reasoning.

order of reference	1	2	3	4	5	6	7	8	9	10	11	12
Type of reference	R	R	W	R	R	W	R	R	W	R	R	W
block referenced	0	15	18	5	1	13	15	26	6	8	15	3

4.3 A system with 350 MHz clock uses a separate data and instruction cache and a unified second-level cache. The first level cache is direct–mapped, write-through, and writes-allocate cache with 8kBytes of data total and 8 Byte blocks, and has a perfect write buffer (never causes and stalls). The first level instruction cache is a direct-mapped cache with 4kBytes of total data and 8 Bytes blocks. The second level cache is a two way set associative, write-back, write-allocate cache with 2Mbytes of total data and 32-Byte blocks.

The 1st level instruction cache has a miss rate of 2%. The first level data cache has a miss rate of 17%. The unified second level cache has a local miss rate of 12%. Assume that 30% of all instructions are data memory accesses; 50% of those are loads and 50% are stores. Assume that 50% of the blocks in the second-level cache are dirty at any time. Assume that there is no optimization for fast reads on L1 or L2 cache miss.

All the first-level cache hits cause no stalls. The second-level hit times is 10 cycles. (That means that the L1 miss penalty, assuming a hit in the L2 cache is, 10 cycles.) Main memory access time is 100 cycles to the first bus width of data; after that, the memory system can deliver consecutive bus widths of data on each following cycle. Outstanding non-consecutive memory requests cannot overlap; an access to one memory location must complete before an access to another location can begin. There is a 128-bit memory to the L2 cache and a 64-bit bus from both the L1 caches to the L2 cache. Assume a perfect TLB for this problem.

a. What percent of all data memory references cause a main memory access (main memory is accessed before the memory request is satisfied)?

b. How many bits are used to index each of the caches? Assume you can use physical addresses for cache.

c. How many cycles can the longest possible data memory access take?

d. What is the average memory access time in cycles (including instruction and data memory references)?

4.4 What are the differences between a write-allocate and no-write-allocate policy in a cache?

4.5 Can a direct mapped cache sometimes have a higher hit rate than a fully associative cache with an LRU replacement policy (on the same reference pattern and with the same cache size)? If so, give an example. If not, explain why not?

4.6 How does a data cache take advantage of spatial locality?

4.7 What is the advantage of using a logically-indexed physically-tagged L1 cache (as opposed to physically-indexed and physically-tagged)? What constraints does this design put on the size of the L1 cache?

4.8 Briefly describe the data access pattern of an application for which a cache with a random replacement policy performs worse than an LRU replacement policy.

4.9 Briefly describe the data access pattern of an application for which a cache with a random replacement policy performs better than an LRU replacement policy.

4.10 Consider a 256kb 4-way set associative cache with 256 byte cache lines for a processor that uses 64-bit data words and 48-bit byte addresses. Assume the variable x, of type uint64, is stored in memory at location 0x4A85_B413_A518. Assume that x is present in the cache, and char* ptr = 0x4A85 B400 0000. Determine if the following accesses will cause a cache miss or hit or say if it can't be determined.
a. *(ptr + 0x11 A538)
b. *(ptr + 0x13 A588)
c. *(ptr + 0x13 0218)

4.11 A computer has a cache, main memory, and a disk used for virtual memory. An access to the cache takes 10 ns. An access to main memory takes 100 ns. An access to the disk takes 10,000 ns. Suppose the cache hit ratio is 0.9 and the main memory hit ratio is 0.8. What is the effective access time (EAT) in ns required to access a referenced word on this system?

4.12 A computer has a memory with 8 GB of address unit and the data bus is 64 bits. If the memory size 256 times larger than cache size and the cache consist 256k block, what would be the block size?

4.13 A processor has 2-way set associative cache. If the number of four-block sets is 32 and memory has 1k times more block than the cache, how many bits does the tag field need?

4.14 The memory access time to one byte of memory is 25 ns. If the cache block size is 16 and cache access time is 2ns, what should be minimum cache hit rate so utilization of cache makes sense?

CHAPTER 5 INTERNAL MEMORY

5.1 The following is an eight bit data word with ECC and one-bit parity: 0100101111010 The bits are in the order

M8M7M6M5M4M3M2M1C8C4C2C1P.

Is this word correct, does it have a one-bit error, or does it have a two-bit error? If it has a one-bit error, correct it.

5.2 Define latency for a main memory bank as the time taken to complete an individual operation (read, write). Suppose a memory system has four banks, each of which has a latency of 100 ns is pipelined to allow 8 operations to overlap execution. If each bank returns 4 bytes of data in response to a memory request, what is the peak throughput in operations/s and the peak data rate in bytes/s of this system?

5.3 Assume a memory system with 2 banks, one of which stores even-addressed words and the other stores odd-addressed words. Assume that the banks have independent connections to the processor, so that there are no conflicts for the memory bus, and that the processor can execute up to 2 memory operations in a given cycle. Also, make the simplifying assumption that the latency of each memory bank is one processor cycle, so the banks are never busy handling requests from previous cycles. Finally, assume that the processor always has 2 memory operations that it would like to execute on a given cycle.

a. What is the peak throughput of the memory system, in operations/cycle?

b. If the addresses of memory requests are random (an unrealistic assumption), how many memory operations will the processor execute each cycle on average?

c. If each memory bank returns 8 bytes of data per request and processor cycles are 10 ns long, what is the peak data rate, in bytes/s, of this memory system and what is its average data rate?

5.4 For each of the following cases, state whether SRAMs or DRAMs would be more appropriate building blocks for the memory system, and state why. Assume that there is only one level of internal memory.

a. A memory system where performance is the most important goal.

b. A memory system where cost is the most important factor.

CHAPTER 6 EXTERNAL MEMORY

6.1 What are the advantages and disadvantages of smaller vs. larger disk sector size?

6.2 Most disks today use a fixed sector size. Many IBM mainframe disks allow the programmer to determine the block (sector) size. Explain advantages and disadvantages of allowing a variable sector size.

6.3 Let us assume a disk with rotational speed of 15,000 rpm, 512 bytes per sector, 400 sectors per track and 1000 tracks on the disk, average seek time is 4ms. We want to transmit a file of size 1 MByte, which is stored contiguously on the disk.

a. What is the transfer time for this file?
b. What is the average access time for this file?
c. What is the rotational delay in this case?
d. What is the total time to read 1 sector?
e. What is the total time to read 1 track?

6.4 A magnetic disk with 5 platters has 2048 tracks/platter, 1024 sectors/track (fixed number of sectors per track), and 512-byte sectors. What is its total capacity?

CHAPTER 7 INPUT/OUTPUT

7.1 Some processors use memory mapped I/O where I/O devices are in the same address space as main memory. Others have separate I/O address space and separate instructions. Give some advantages and disadvantages of each.

7.2 Although DMA does not use the CPU, the maximum transfer rate is still limited. Consider reading a block from the disk. Name three factors that might ultimately limit the rate transfer.

7.3 **Spooling** is an I/O technique developed in the early days of multiprogramming systems. It is a way of dealing with a dedicated I/O device. Each user process generates an I/O stream for a device that is directed to a disk file rather then to the device itself. The responsibility for moving data between the disk and the required device is delegated to a separate process, called a spooler. Spooling is appropriate for devices that cannot accept interleaved data streams, such as a printer.
Prefetching is a method of overlapping the I/O of a job with that job's own computation. The idea is simple. After a read operation completes and the job is about to start operating on the data, the input device is instructed to begin the next read immediately. The processor and input device are then both busy. With luck, by the time that the job is ready for the next data item, the input device will have finished reading that data item. The processor can then begin processing the newly read data, while the input device starts to read the following data. A similar idea can be used for output. In this case, the job creates data into a buffer until an output device can accept them.
Compare the prefetching scheme with the spooling scheme, where the processor overlaps the input of one job with the computation and output of other jobs.

7.4 A given processor requires 1000 cycles to perform a context switch and start an interrupt handler (and the same number of cycles to switch back to the program that was running when the interrupt occurred), or 500 cycles to poll an I/O device. An I/O device attached to that processor makes 150 requests per second, each of which takes 10,000 cycles to resolve once the handler has been started. By default , the processor polls every 0.5 ms if it is not using interrupts.

a. How many cycles per second does the processor spend handling I/O from the device if interrupts are used?

b. How many cycles per second are spent on I/O if polling is used (include all polling attempts)? Assume the processor only polls during time slices when user programs are not running, so do not include any context-switch time in your calculation.

c. How often would the processor have to poll for polling to take as many cycles per second as interrupts?

7.5 An I/O device transfers 10 MB/s of data into the memory of a processor over the I/O bus, which has a total data transfer capacity of 100 MB/s. The 10 MB/s of data is transferred as 2500 independent pages of 4 KB each. If the processor operates at 200 MHz, it takes 1000 cycles to initiate a DMA transaction, and 1500 cycles to respond to the device's interrupt when the DMA transfer completes, what fraction of the processor's time is spent handling the data transfer with and without DMA?

7.6 If the normal processor memory read (RD) and write (WR) control outputs are connected to I/O interface adapters, what type of I/O technique is being used?

CHAPTER 8 OPERATING SYSTEM SUPPORT

8.1 What is the primary advantage of adding a translation lookaside buffer (TLB)?

8.2 Suppose you have a computer system with the following characteristics: The virtual memory system uses a 4K page size. Its TLB has 16 sets (we don't need to know how associative it is, nor anything else about the virtual memory system, for this question). The first-level cache subsystem is a 32K, 8-way set-associative cache with a 16 byte line size. It uses physical addresses. An attempt is made to access address 0xabcd1234.

a. What will the page offset field for this address contain?
b. Which set in the TLB will this address map to?
c. What will the TLB tag have to be for us to have a TLB hit?
d. What will the address's cache offset field contain?
e. Which set in the cache will this address map to?
f. Assume we had a TLB hit in part (b), and that the page frame number is 0x1247. What would the cache tag have to be for us to have a cache hit?
g. Would it have been possible for hardware to solve parts (a) and (c) in parallel, and parts (b) and (d) in parallel (i.e., to perform the TLB lookup and cache lookup simultaneously)?

8.3 Give two ways virtual memory address translation is useful even if the total size of virtual memory (summed over all programs) is guaranteed to be smaller than physical memory.

8.4 Consider a processor with a 32-bit virtual address and a 32-bit physical address.

Level 1 Cache: 64Kbyte, 16 byte cache line size, 1% miss rate, 1 ns access time
Level 2Cache: None
RAM: 15 ns access time
TLB: 4 kByte page size
The Level-1 cache and TLB are accessed in parallel.

Note: The location of instructions and data in the cache is a function of the n low-order bits of the address of the data or instruction. Consequently, instructions and data that have the same n low-order bits compete for the same cache space. Such competing objects are called

cache aliases of each other and this technique of organizing data/instruction is called cache aliasing.

a. Is the L-1 cache addressed with virtual or physical addresses?
b. For the L-1 cache, how big is the offset, index, and tag fields?
c. For the TLB, how big is the virtual page number (VPN)?
d. How many bits of address are needed to resolve cache aliasing?

8.5 The classical batch processing system ignores the cost of increased waiting time for users. Consider a single batch characterized by the following parameters:

M average mounting time
T average service time per job
N number of jobs
S unit price of service time
W unit price of waiting time per user

a. Show that the optimal batch size that minimizes the cost of service time and waiting time per user (within a single batch) is

$$N_{opt} = \sqrt{\frac{M}{T}\frac{S}{W}}$$

b. In an installation in which M = 5 min, T = 1 min, and S = $300/hr, the operators choose N = 50. Assuming that this is an optimal choice, find the unit cost of the user's waiting time W.

8.6 In the Exec II batch system, users would submit a large number of jobs in the morning. These jobs took hours to complete and thereby prevented fast response. Suggest a modification of the scheduling policy that would discourage users from doing this.

8.7 In the Scope system for the CDC 6600 computer, system resources (processor time, storage, etc.) can remain idle while running jobs wait for operators to mount magnetic tapes. Suggest a solution to this problem.

8.8 Measurements on the CTSS system showed that about half of all user requests could be classified as file manipulation, program input, and editing. How would you use this information about expected workload to improve processor utilization at a reasonable cost without degrading user response?

CHAPTER 9 NUMBER SYSTEMS

9.1 Convert the following numbers to base 10:

a. 3124_7
b. 12332_4
c. 15366_9
d. 4000_5
e. 7122_8
f. 77_8
g. 333_4
h. 111111_2

9.2 Convert the following base 10 numbers to the indicated base:

a. 413 to base 5
b. 128 to base 8
c. 3000 to base 6
d. 963 to base 3
f. 67 to base 2

9.3 Normal trinary (base 3) uses the digits 0, 1, and 2 with place values being powers of three. In symmetric trinary the digits have the values zero, one, and negative one. The symbol $\perp$ will be used to represent negative one, though more traditionally it is represented as a one with an overbar. Symmetric trinary allows representing positive and negative values without an explicit sign. The place values are still powers of three, such as 243, 81, 27, 9, 3, and 1. Since the largest digit value is 1, half that in normal trinary, the largest value that can be represented in a given number of digits is also half as large.

Represent the following decimal numbers in symmetric trinary:

a. 23
b. 99
c. –23

9.4 Count from 0 to decimal 20 in symmetric trinary.

CHAPTER 10 COMPUTER ARITHMETIC

10.1 Convert the decimal number –30.375 to IEEE 754 floating-point format. Show the result in both binary and hexadecimal.

10.2 Multiply two single-precision floating-point numbers and write single-precision floating-point product C=A × B. The numbers are in IEEE 754 format. Show the results in decimal and 754 format.
A=0x3FE00000, B=0xC0800000

10.3 The nines complement of a decimal number is formed by subtracting each digit from nine. Show:
- **a.** The nines complement of 123. (That is, the representation of –123.)
- **b.** The nines complement of zero in thee digit arithmetic
- **c.** The nines complement of 777

10.4 The floating point format for the PDP-10 processor has in interesting feature: While positive numbers are represented similar to the way most processors represent binary floating point value, negative values are formed by taking the twos complement of the entire 36 bit word. What advantages and disadvantages does this representation have?

10.5 Translate the following two complement numbers into their signed decimal equivalents.
a. 11111011 **b.** 00001111 **c.** 10001111 **d.** 01110111

10.6 Add the following signed decimals using twos complement numbers. Show your work.
a. (+1) + (–6) **b.** (+20) + (–60) **c.** (+8) + (–5) **d.** (+89) + (–46)

In the following four problems, assume the computer truncates the significand to four decimal digits and show your results as normalized decimal floating-point numbers.

10.7 Perform the following floating-point additions:
- **a.** $(0.2233 \times 10^{2}) + (0.6688 \times 10^{1})$
- **b.** $(5.666 \times 10^{0}) + (44.55 \times 10^{0})$
- **c.** $(111.77 \times 10^{0}) + (55.666 \times 10^{0})$

10.8 Perform the following floating-point subtractions:
a. $(0.9922 \times 10^{-3}) - (0.4477 \times 10^{-3})$

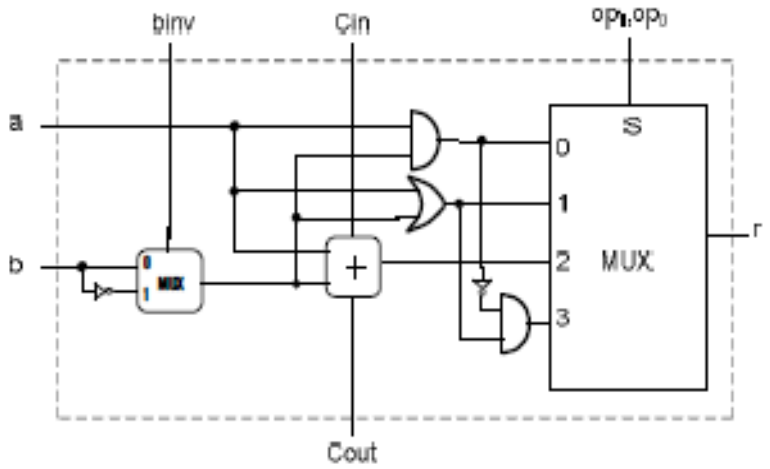
b. $(33.666 \times 10^{0}) - (2.7777 \times 10^{0})$
c. $(0.8888 \times 10^{2}) - (0.2222 \times 10^{3})$

10.9 Perform the following floating-point multiplications:
a. $(0.5432 \times 10^{3}) \times (0.3333 \times 10^{-5})$
b. $(222.88 \times 10^{0}) \times (1.1177 \times 10^{0})$

10.10 Perform the following floating-point divisions:
a. $(0.2233 \times 10^{-2})/(0.6611 \times 10^{3})$
b. $(111.99 \times 10^{0})/(44.888 \times 10^{0})$

CHAPTER 11 DIGITAL LOGIC

11.1 Consider the following 1-bit ALU and find the ALU functions



CHAPTER 12 INSTRUCTION SETS: CHARACTERISTICS AND FUNCTIONS

12.1 We store the following 4 instructions in memory in the original order, please fill the memory contents (in hex) below for (1) a Little Endian system; (2) a Big Endian system. Depict the memory as a column of boxes, where each box represents a single byte.)

		Encoding
0x10010000:	addi $10, $9, -2	0x212afffe
	sra $16, $12, 1	0x000c8043
	divu $12, $20	0x0194001b
	bne $15,$0, L2	0x15e00005

12.2 The 3-bit hex value 30A79847 is stored in location 1000 in a byte-addressable machine. What is the value of the byte in address 1002 if the system is
a. big endian?
b. little endian?

12.3 Compute the value of the C expression ('z' – 'a'), where ASCII coding is used.

12.4 Right arithmetic shift and right logical shift correspond to divide by two (or powers of two). Compare the results to the usual implementation of signed and unsigned divide for both positive and (twos complement) negative values.

12.5 Just about all machines now use twos complement representation for negative numbers, but ones complement machines are still available, and were less unusual in times past. Compute the results of right arithmetic shift in ones complement for:
a. decimal –5
b. decimal –6

12.6 The *r*'s complement of an *n*-digit number *N* in base *r* is defined as $r^n - N$ for $N \neq 0$ and 0 for $N = 0$. Compute 456 – 123 using nines complement (decimal) arithmetic.
(That is, 456 + (–123) representing –123 in nines complement form.

CHAPTER 13 INSTRUCTION SETS: ADDRESSING MODES AND FORMATS

13.1 Suppose that on a MIPS system, registers $10 and $15 contains the following 32-bit binary data

$10= 1111 1111 1111 1111 1111 1111 1001 0110 = −106 signed or 4294967190 unsigned
$15= 0000 0000 0000 0000 0000 0000 0001 0001 = 17

What is the value of register $20 after execution of:
a. sra $20, $10, 5
b. lui $20, 20
c. srl $20, $10, 3
d. sltu $20, $10, $15
e. sltiu $20, $10, 20
f. slti $20, $10, −20
g. slt $20, $10, $15
Note: please refer to MIPS_Instruction_Reference.pdf in the Useful Documents section of this book's Web site.

13.2 Hand-assemble the following (Mini-)MIPS assembly-language code fragment to correct machine language. Fill in your answer in binary in two rows of 32 bits each. Also, annotate your answer to clearly delineate all of the fields of each instruction.

Also, indicate the format type of each instruction (I, J, R) by writing the appropriate letter to the left of its row.

Finally, write out the corresponding machine-language code as a pair of hexadecimal words:

```
add   $t1, $s0, $t0
lw    $t2, -4($t1)
```

Note: please refer to MIPS_Instruction_Reference.pdf in the Useful Documents section of this book's Web site.

13.3 There exists processor with 16-bit word length (8-bit addressing unit) with a memory size of 32k words. The addressing method is register-based and direct. Instructions are encoded in two types: single-word and two-word with 3 addresses. Assuming there are no more than 576

different instructions, what is the maximum possible number of general purpose registers in this processor?
Note: In single-word instructions all three operands are registers.

13.4 In a single-address processor, immediate and direct addressing methods are used. If the ALU output is 8 bits and memory size is 4 kB, what would be the maximum possible number of instructions in this processor?

CHAPTER 14 PROCESSOR STRUCTURE AND FUNCTION

14.1 Consider an unpipelined processor. Assume that it has 1-ns clock cycle and that it uses 4 cycles for ALU operations and 5 cycles for branches and 4 cycles for memory operations. Assume that the relative frequencies of these operations are 50%, 35% and 15% respectively. Suppose that due to clock skew and set up, pipelining the processor adds 0.15 ns of overhead to the clock. Ignoring any latency impact, how much speed up in the instruction execution rate will we gain from a pipeline?

14.2 What is the relaxed consistency model in context of register ordering? You will need to do research beyond the book to answer this question.

14.3 Explain what control hazard is and how to avoid it.

14.4 Assume we are working with an architecture where it is "easy" to identify call and return instructions (as distinct from other branch instructions or non-branch instructions). Assume also that the BTB always hits and provides accurate information about the instruction being fetched. The return address stack (RAS) is a small structure that contains an array of (say) 8 addresses and a "head pointer" that points to one entry of the array. Each time we speculatively fetch a call instruction we push the expected return point of the call (PC + 4) onto the RAS. We do this by incrementing the head pointer (mod 8) and writing the return address at the corresponding RAS array location. Each time we speculatively fetch a return instruction we predict that the next PC will be the PC currently pointed to by the head pointer. Then we decrement the head pointer (mod 8).

a. First, assume that we don't do any "repair" of the RAS after a branch misprediction. Give a (short) sequence of instructions and events that will cause the RAS to get out of sync (i.e., mispredict all the return addresses currently listed in the RAS).

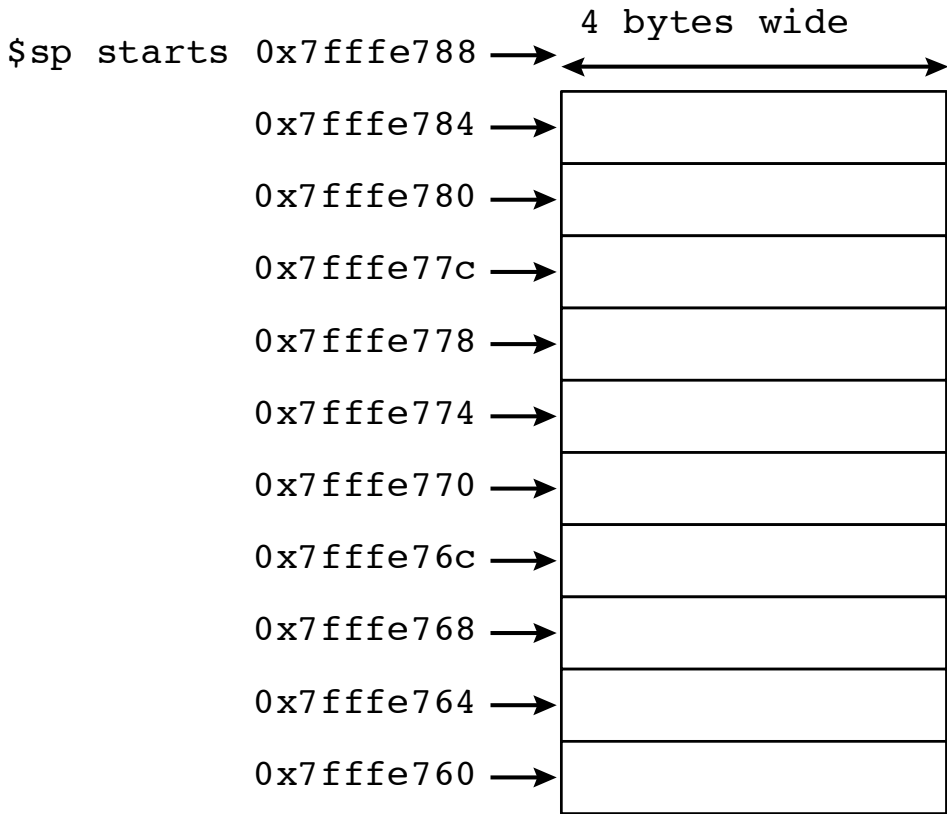
b. Now, assume that we repair the state of the RAS as follows: with each branch instruction we carry down the pipeline the value of the RAS head pointer, as it was when the branch was fetched. (Much the same as we do with the global history). When a branch misprediction is detected, we reset the RAS head pointer to the position we carried with the mispredicted branch. Give a (short) sequence of instructions and events that will cause at least the next

return address to be mispredicted, but not the contents of the rest of the stack.

14.5 Given the MIPS code as shown below. (Assume that there is no delay slot.), note that the stack pointer ($sp or $29) initially points at 0x7fffe788 and grows toward lower address. Please answer the following questions.

```
                            # $sp = 0x7fffe788;
0x4001000:                  addiu  $16,$0,4
                            add    $4,$0,$16
                            jal    function
                            add    $a0, $2, $0
                            li     $v0, 1
0x4001014:                  syscall              # syscall to print number
                            ...
0x400f000:    function:     addi   $sp,$sp,-8
                            sw     $16,($sp)
                            add    $16,$0,$4
                            slt    $2,$16,2
                            sw     $31,4($sp)
                            beq    $2,$0,L20
                            add    $2,$0,$16
                            j      L22
0x400f020:          L20:    addi   $4,$16,-1
                            jal    function
                            add    $2,$16,$2
                    L22:    lw     $31,4($sp)
                            lw     $16,($sp)
                            addiu  $sp,$sp,8
                            j      $31
```

a. Fill up the contents of the stack memory below when the program reaches label “L22”. You might not need all the blank boxes. You want to keep track the register contents, at least register $2, in order to answer question.



b. What is the value stored inside register $a0 when you finally return to the syscall instruction, i.e. PC address = 0x4001014?

Note: please refer to MIPS_Instruction_Reference.pdf in the Useful Documents section of this book's Web site.

14.6 Consider a processor with the following two-level branch predictor. The first level predictor is a one-bit predictor, which always predicts a branch will be taken if it was taken on the last execution of the instruction. The first level predictor is initialized to branch-taken. Branches correctly predicted by this predictor have 0 cost. The second level predictor is a two-bit predictor using a counter; its states are

00=strongly don't branch,
01=weakly don't branch,
10=weakly branch, and
11=strongly branch.

This predictor is initialized to state 10. Predictions changed by this predictor will incur a 1 cycle cost. Finally, corrections made by the actual branch evaluation unit have a 2 cycle cost. These costs are not cumulative: if the first level predictor makes a correct prediction, the second level predictor incorrectly "corrects" it, but the branch evaluation unit validates the first level predictor, there is no cost. Now consider the following code:

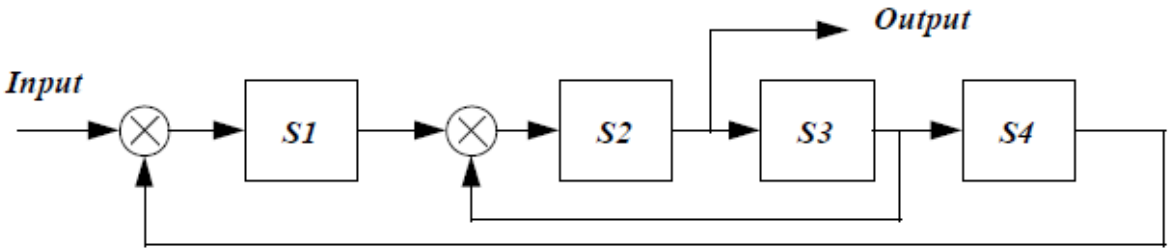
```
i = 0;
do {
 j = 0;
 do {
      j = j + 1;
 } while (j < 10);
 i = i + 1;
     } while (i < 10);
```

a. What is the mean branch cost for the branch at the end of the outer loop?

b. What is the mean branch cost for the branch at the end of the inner loop?

c. What is the average branch cost for all branches in the program?

14.7 Consider the following pipelined processor with four stages. This pipeline has a total evaluation time of six clock cycles. All successor stages must be used after each clock cycle.



a. Specify the reservation table for this pipeline with six columns and four rows. (reservation tables are discussed in Appendix I)

b. List the set of forbidden latencies between task initiations.

c. List all greedy cycles from the state diagram (greedy cycles are discussed in Appendix I).

d. Determine the minimal average latency (MAL)

14.8 Consider the following pseudo-code snippet:
(address in hexadecimal, followed by pseudo assembly code)

```
1000: BRANCH <condA> 1018
1004: BRANCH <condB> 1014
1008: BRANCH <condC> 1014
100C: ADD
1010: MUL
1014: BRANCH <condE> 1004
1018: SUB
```

Suppose we are using a gshare predictor with a history of length 2 (just to keep the example simple). Suppose also that we have a 32 entry table of counters. Our gshare predictor hashes into this table by taking bits [6:2] of the current PC, and xoring with the current 2-bit history.)

NOTE: A gshare predictor is a more advanced dynamic branch predictor than the regular binomial predictor that uses the history of recently executed branches to predict the next branch. Gshare uses a history register to records the taken/not-taken history of the last *h* branches. It does this by shifting in the most recent conditional branch outcome into the low-order bits of the branch history register. It then hashes the branch history and the PC of the branch when making predictions. Specifically, gshare uses the lowest *n* of the xor of the program counter and the history register.

a. Construct two paths through the code, ending at different branch instructions, but that cause the predictor to hash to the same table entry. (This is probably easier if you draw the code out as a flow graph.

b. Construct two distinct legal paths through the code, both ending at the BRANCH at address 1014, but that produce the same 2 bit global history.

14.9 Suppose that we measure the following instruction mix for a program:

Loads: 25%, Stores: 15%, Integer: 30%, Floating-Point: 20%, Branches: 10%

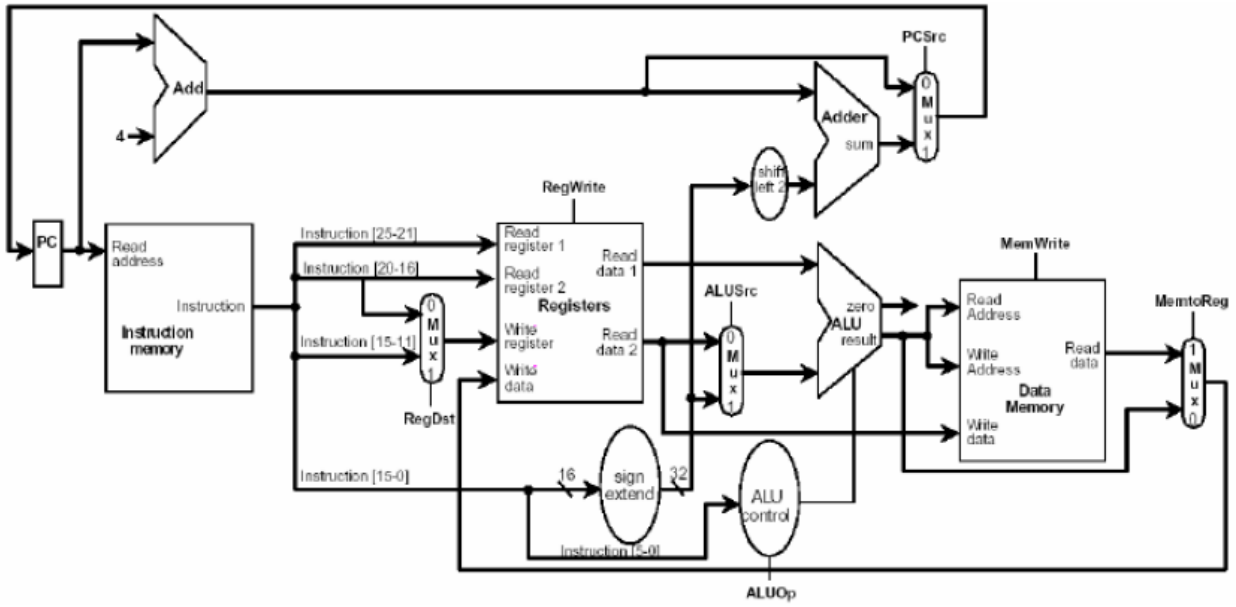
Assume that the processor is using the 5-stage pipeline (base CPI of 1.0). Data hazards cause an average penalty of 0.9 cycles for floating point operations. Integer operations run at maximum throughput. The processor uses the predict-branch-not-taken technique, which turns out to be correct for 80% of the branches. For the remaining branches, there is a 1 cycle stall. What is the average CPI of this program including memory misses (see the table)? Assume that stores have the same average memory access time (AMAT) as loads

	Hit Time	Miss Rate	Block Size
Level-1 cache	1 cycle	6% for data 2% for instruction	32 bytes
Level-2 cache	12 cycles + (1 cycle per 64 bits)	2%	256 bytes
DRAM	70ns + (10ns per 8 bytes)	–	–

CHAPTER 15 REDUCED INSTRUCTION SET COMPUTERS

15.1 A processor has a non-linear pipeline with 4 stages A, B, C and D. Each instruction goes through different stages in the following order A B C B A D C. Find the bounds on the maximum instruction throughput in a static hazard free schedule.

15.2 The instruction, sw $18, -12($29), which stores a word (4 bytes) from register 18 into the memory location with an address equal to the contents of register 29 minus the value 12, is executed on the MIPS single cycle datapath. Assume the registers are all pre-loaded with the register number prior to execution and each data memory byte is loaded with the least significant byte of its own address. For example, 0x00000001 contains 0x01, 0x00000002 contains 0x02, 0x0000003F contains 0x0F, etc. Thus, if you lw (load 4 bytes) from 0x00000000, you will get 0x03020100; lw from 0x00000008 will get 0x0B0A0908. Note that little endian is used. Fill in the values of the following buses and control signals in Hex. (You must place an "X" in the blank if the signal is a "don't care".)



Buses
Instruction[15-0] after sign extension = ?
Read data 1 = ?
Read data 2 = ?
Write Address in Data Memory = ?

Control signals
ALUSrc = ? MemtoReg = ?
PCSrc = ? MemWrite = ?
RegDst = ? RegWrite = ?

15.3 a. Identify all data dependencies in the following code, assuming that we are using the 5-stage MIPS pipelined datapath. Which dependencies can be resolved via forwarding?

```
add $2,$5,$4
add $4,$2,$5
sw $5,100($2)
add $3,$2,$4
```

b. Consider executing the following code on the 5-stage pipelined datapath: Which registers are read during the fifth clock cycle, and which registers are written at the end of the fifth clock cycle? Consider only the registers in the register file (i.e., $1, $2, $3, etc.)

```
add $1,$2,$3
add $4,$5,$6
add $7,$8,$9
add $10,$11,$12
add $13,$14,$15
```

15.4 You are designing the new Illin 511 processor, a 2-wide in-order pipeline that will run at 2GHz. The pipeline front end (address generation, fetch, decode, and in-order issue) is 14 cycles deep. Branch instructions then take 6 cycles to execute in the back end, load and floating-point instructions take 8 cycles, and integer ALU operations take 4 cycles to execute.

a. Your lab partner has written some excellent assembly code that would be able to achieve a sustained throughput on the Illin 511 of 4 billion instructions/second, as long as no branches mispredict. Assume that an average of 1 out of 10 instructions is a branch, and that branches are correctly predicted at a rate of *p*. Give an expression for the average sustained throughput in terms of *p*.

b. Unfortunately, it turns out that there was a bug in the original code. The bug fix made the code somewhat larger, and now you are observing instruction cache misses in the 2Kbyte L1 instruction cache. Each L1 cache miss causes a 10 cycle bubble to be inserted in the instruction stream (while we fetch the cache line from the L2). If you could only double the size of the L1 cache you could completely eliminate the L1 cache misses. Unfortunately building a 4Kbyte fully pipelined2GHz L1 cache would add an extra 5 cycles to the front end. Under what circumstances (e.g. cache miss rate and branch prediction rate) would it be a good or bad idea to change the pipeline to include the larger cache?

15.5 Briefly give two ways in which loop unrolling can increase performance and one in which it can decrease performance.

15.6 Some RISC architectures require the compiler (or assembly programmer) to guarantee that a register not be accessed for a given number of cycles after it is loaded from memory. Give an advantage and a disadvantage of this design choice.

15.7 Consider the following code:

```
Loop:lw $1, 0($2)
addi $1, $1, 1
sw $1, 0($2)
addi $2, $2, 4
sub $4, $3, $2
bne $4, $0, Loop
```

Assume that the initial value of R3 is R2 + 396

This code snippet will be executed on a MIPS pipelined processor with a 5-stage pipeline. Branches are resolved in the decode stage and do not have delay slots. All memory accesses take 1 clock cycle. In the following three parts, you will be filling out pipeline diagrams for the above code sequence. Please use acronyms F, D, X, M and W for the 5 pipeline stages. Simulate at most 7 instructions, making one pass through the loop and performing the first instruction a second time.

a. Fill in a pipeline diagram in the style of Figure 15.7 for the execution of the above code sequence without any forwarding or bypassing hardware but assuming a register read and a write in the same clock cycle "forwards" through the register file.

b. Fill in a pipeline diagram for the execution of the above code sequence with traditional pipeline forwarding:

CHAPTER 16 INSTRUCTION-LEVEL PARALLELISM AND SUPERSCALAR PROCESSORS

16.1 A processor with dynamic scheduling and issue bound operand fetch has 3 execution units – one LOAD/STORE unit, one ADD/SUB unit and one MUL/DIV unit. It has a reservation station with 1 slot per execution unit and a single register file. Starting with the following instruction sequence in the instruction fetch buffer and empty reservation stations, for each instruction find the cycle in which it will be issued and the cycle in which it will write result.

```
load R6, 34(R12)
load R2, 45(R13)
mul R0, R2, R4
sub R8, R2, R6
div R10, R0, R6
add R6, R8, R2
```

Assume out of order issue and out of order execution. Execute cycles taken by different instructions are:

LOAD/STORE : 2
ADD/SUB : 1
MUL : 2
DIV : 4

16.2 State whether the following are true or false and also give the reason for your answer

a. One of the main benefits of scoreboarding the CPU is the reduction of WAR and WAW hazards through register renaming.

b. As a rule of thumb supported by SPECmark benchmarking, if an m-bit, k-entry branch prediction buffer (BPB) is able to reduce misprediction frequency by a factor of x, then a $2m$-bit, $2k$-entry BPB will reduce misprediction frequency by a factor of close to $4x$.

16.3 Which of the following variations of cache technology is strongly associated with superscalar processing?

1. "hit under miss"
2. segregated caches
3. multiported caches
4. subblock placement
5. high associativity

16.4 You are designing the issue unit for a 1-wide pipeline with in-order issue and out-of-order completion. In this pipeline, once an instruction issues, it reads its source register values on the first cycle and then proceeds to the ALU on the following cycle. Branch instructions take 6 cycles to execute in the back end, load and floating-point instructions take 8 cycles, and integer ALU operations take 4 cycles to execute (including the cycle to read the registers). Instructions write to the register file on the cycle they complete. Branch mispredictions are detected 6 cycles after the branch issues. On a mispredict all preceding pipeline stages are flushed. Luckily, none of the instructions can ever take an exception. (So you don't need to support precise exceptions).

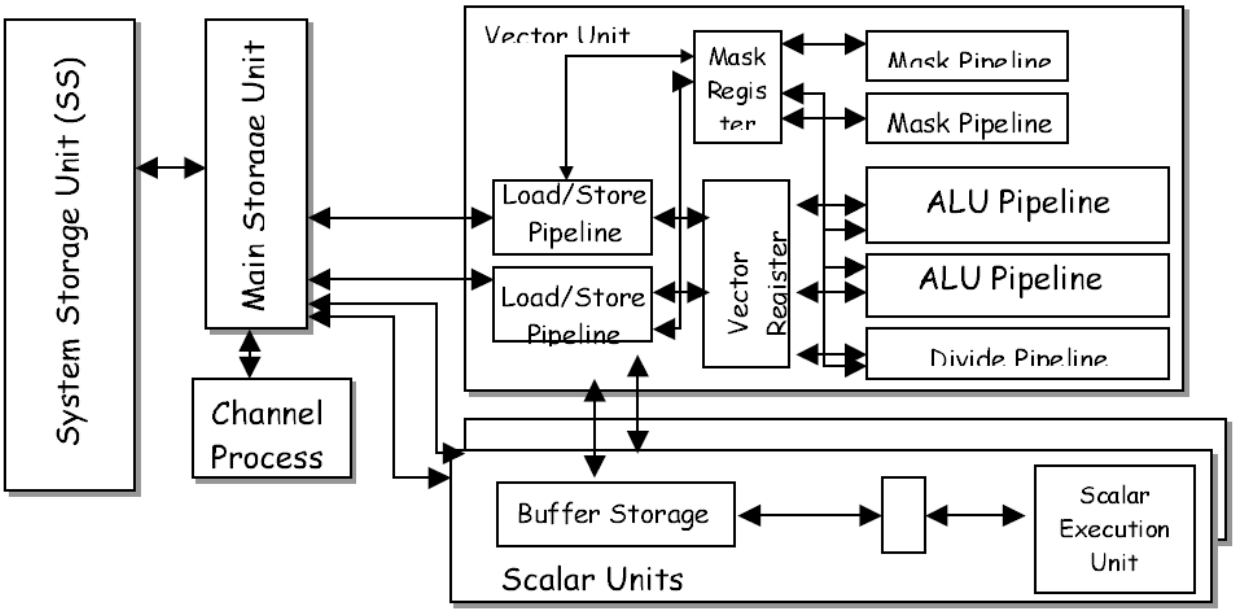
a. One of the rules that the instruction issue unit must follow is: an instruction must wait at the issue unit for its source operands to be written/completed before the instruction can issue. Since this is an in-order processor the instruction issue unit also follows the rule: an instruction must wait at the issue unit for all preceding instructions to issue. Give the rest of the rules that the issue unit must follow.

b. Describe two ways in which implementing renaming could improve this designs performance (even if we keep it an in-order issue design).

CHAPTER 17 PARALLEL PROCESSING

17.1 Vectorizing compilers generally detect loops that can be executed on a pipelined vector computer. Are the vectorization algorithms used by vectorizing compilers suitable for MIMD machine parallelization? Justify your answer.

17.2 On a Fujitsu VP2000, the vector processing unit is equipped with two load/store pipelines plus five functional pipelines as shown below.



Consider the execution of the following compound vector function (CVF):

$$A(I) = B(I) \times C(I) + D(I) \times E(I) + F(I) \times G(I)$$

for I=1, 2, ..., N. Initially, all vector operands are in memory, and the final vector result must be store in memory. Show the space-time diagram, for pipelined execution of the CVF. Note that, two vector loads can be carried out simultaneously on the two vector access pipes. At the end of computation, one of the two access pipes is used for storing the A array.

17.3 Figure 1 shows state diagrams for two different variations on an MSI snoopy cache coherence protocol.

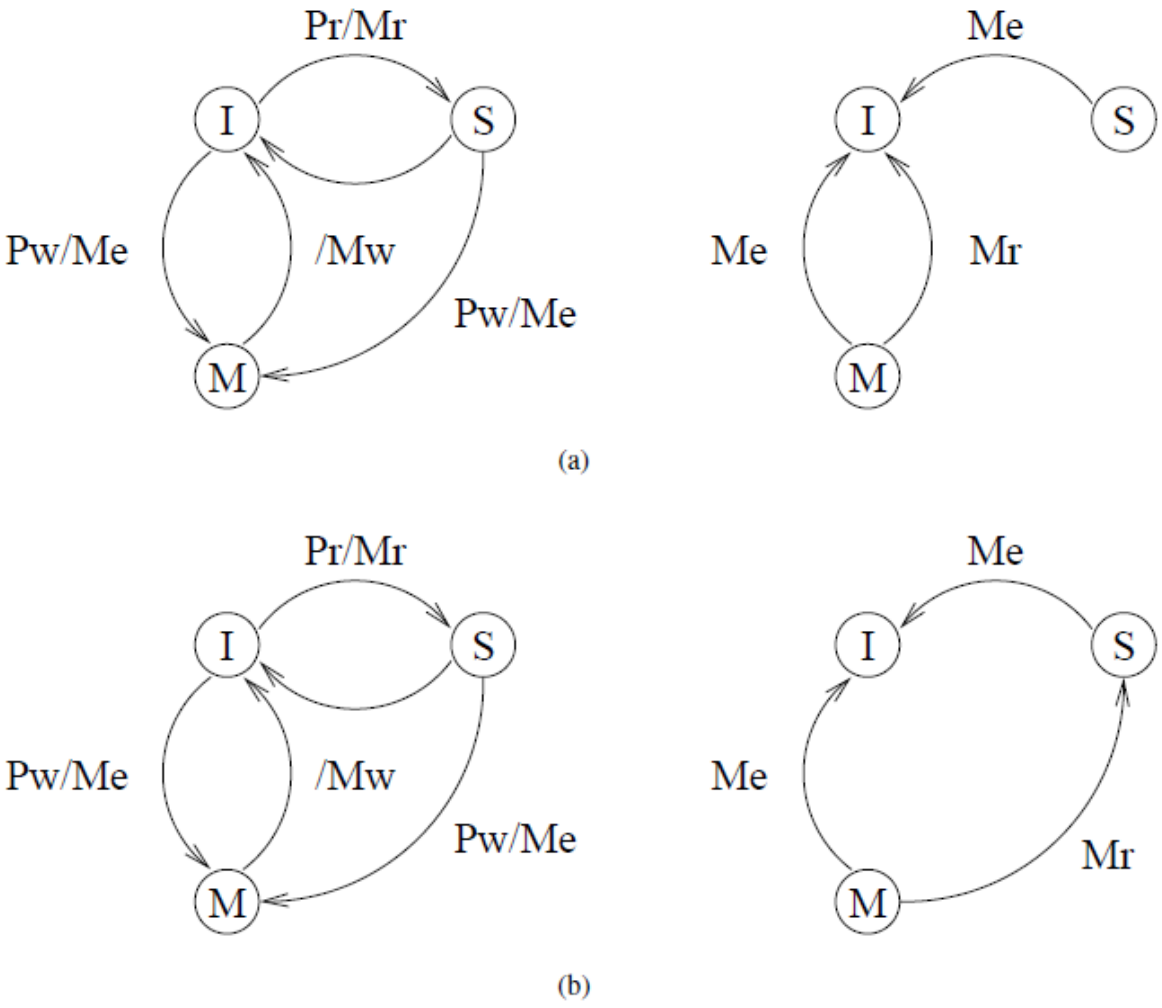


Figure 1: Cache Coherence State Transitions

For clarity, the figures show state transitions in response to CPU operations separately from transitions in response to snooped memory operations. The responses to CPU operations are on the left, and the responses to memory operations are on the right. The two CPU operations are Pr (processor read) and Pw (processor write). The three memory operations are Mr (memory read), Mw (memory write), and Me (memory read exclusive). The transitions show the state change that occurs in response to processor (or memory) operations, and the memory operations that may occur as a result. For instance, the arc from state I to state S labeled Pr/Mr in figure 1(a) says that if a cache block is Invalid, and the processor executes a read operation, the cache block will change to state S and a memory read operation will occur as a result. The arcs with no processor operation are cache flushes. A block in state S can transition to state I without writing to memory (so these arcs don't show any label at all); a block in state M must write its data to memory if it is to transition to state I.

a. Suppose processor P1 performs a test-and-set-bit instruction. In this instruction, the processor reads a memory location to determine its old contents, and then writes new contents to it. If the memory location was previously not valid in any cache block, what will its state be in P1's cache block following the operation?

b. Now, following the operation in part (a), suppose P2 performs the same instruction on the same memory location. What will the block's state be in the two processor's caches now? Will the variation chosen make any difference to the final state?

17.4 Imagine a dual processor machine with CPUs A and B. Explain the difficulty of CPU A performing fetch-and-increment(x) when the most recent copy of x is cleanExclusive [the data is not cached (or in other words is most recent) before and is mutually exclusive with respect to B] in CPU B's cache. You may wish to illustrate the problem with a short sequence of events at processor A and B. Note: Here the states defined for this problem.

- Invalid (I): Block is not present in the cache.
- Clean exclusive (CE): The cached data is consistent with memory, and no other cache has it.
- Owned exclusive (OE): The cached data is different from memory, and no other cache has it. This cache is responsible for supplying this data instead of memory when other caches request copies of this data.
- Clean shared (CS): The data has not been modified by the corresponding CPU since cached. Multiple CS copies and at most one OS copy of the same data could exist.
- Owned shared (OS): The data is different from memory. Other CS copies of the same data could exist. This cache is responsible for supplying this data instead of memory when other caches request copies of this data. (Note, this state can only be entered from the OE state.)
- Coherent Read (CR): issued by a cache on a read miss to load a cache line.
- Coherent Read and Invalidate (CRI): issued by a cache on a write-allocate after a write miss.
- Coherent Invalidate (CI): issued by a cache on a write hit to a block that is in one of the shared states.

CHAPTER 18 MULTICORE COMPUTERS

18.1 In this question we will analyze the performance of the following C program on a multithreaded architecture. You should assume that arrays A, B and C do not overlap in memory.

C code

```
for (i=0; i<328; i++) {
    A[i] = A[i] * B[i];
    C[i] = C[i] + A[i];
}
```

Our machine is a single-issue, in-order processor. It switches to a different thread every cycle using fixed round robin scheduling. Each of the N threads executes one instruction every N cycles. We allocate the code to the threads such that every thread executes every Nth iteration of the original C code.

Integer instructions take 1 cycle to execute, floating point instructions take 4 cycles and memory instructions take 3 cycles. All execution units are fully pipelined. If an instruction cannot issue because its data is not yet available, it inserts a bubble into the pipeline, and retries after N cycles.

Below is our program in assembly code for this machine for a single thread executing the entire loop.

```
loop:  ld f1, 0(r1)      ; f1 = A[i]
       ld f2, 0(r2)      ; f2 = B[i]
       fmul f4, f2, f1   ; f4 = f1 * f2
       st f4, 0(r1)      ; A[i] = f4
       ld f3, 0(r3)      ; f3 = C[i]
       fadd f5, f4, f3   ; f5 = f4 + f3
       st f5, 0(r3)      ; C[i] = f5
       add r1, r1, 4     ; i++
       add r2, r2, 4
       add r3, r3, 4
       add r4, r4, -1
       bnez r4, loop     ; loop
```

a. We allocate the assembly code of the loop to N threads such that every thread executes every Nth iteration of the original loop. Write the assembly code that one of the N threads would execute on this multithreaded machine.

b. What is the minimum number of threads this machine needs to remain fully utilized issuing an instruction every cycle for our program?

c. Could we reach peak performance running this program using fewer threads by rearranging the instructions? Explain briefly.

d. What will be the peak performance in flops/cycle for this program?

CHAPTER 20 CONTROL UNIT OPERATION

20.1 Consider the following pseudo assembly code for computing c = a + b. Assume that a, b, and c are assigned to consecutive memory "words" (memory is generally addressed byte by byte and assume that a word is 4 bytes) and address for "a" is 0x0000ec00. Also, we have a = 22, b = 158, and c = 0 at the starting time. Assume that the first instruction of the code is stored in 0x0000b128. Also, each instruction has the opcode in the first byte (most significant byte) and the remaining 3 bytes specify the corresponding address. The opcode for store is 1, load is 2, and add is 3.

0x0000b128 load a
0x0000b12c add b
0x0000b130 store c

a. Show the memory addresses and contents for all the instructions and data involved. Use the format as follows to express your answer (but the following is not the answer). For all data, use hexadecimal representation.

addresses	contents
0x00002104	0x00000001
0x00002108	0x00000002
……	……

b. Write the micro instructions for the code segment. Assume that current PC (program counter) is 0x00001018. For each micro-instruction, also indicate the data that is transferred (if it is a memory access). For all data, use the hexadecimal representation. The following are the first two micro-instructions and the data transferred. Complete the rest.

Micro-instructions	data
PC → MAR	0x0000b128
M → MBR	0x0200ec00

APPENDIX B ASSEMBLY LANGUAGE AND RELATED TOPICS

B.1 The following C language function computes the integer square root, that is, given an integer n ≥ 0, it returns the value r of the largest integer that is less than or equal to the square root of n, or, in symbols,

$$r = \lfloor \sqrt{n} \rfloor$$

Design a correct implementation of this function in MIPS assembly language; that is, hand-compile this C code to equivalent MIPS assembly. Use standard MIPS calling conventions, except don't worry about the frame pointer. Assume that an int is 32 bits. Don't worry about whether this particular algorithm does anything sensible when the argument is negative (it doesn't). It's also not very efficient, but it works.

```
int sqrt(int n){     /* Find integer square root of n*/
int r=0;             /* First trial square root = 0 */
while (r*r <= n)/* While it's not too big, */
 r++;           /*   keep making it bigger. */
return r-1;     /* Too big, return previous. */
}
```

Comment your code, and draw a table showing any correspondences that you may have set up between MIPS registers and C code variables Note: please refer to MIPS_Instruction_Reference.pdf in the Useful Documents section of this book's Web site.

B.2 The following C language function intpower() computes integer powers, that is, given an integer *b*, and an integer exponent *e* ≥ 0, it returns the value of b^e, if this value fits in an int. (If not, then the result will be meaningless.)

```
int intpower(int b, int e){   /* Find b to the power e.      */
    int r = 1;             /* Initialize result to 1.    */
    while (e>0) {         /* While exponent's more than 0, */
     r *= b;         /*   multiply result by base,    */
     e--; }           /*   and decrement exponent.     */
    return r;             /* Return result.               */
}
```

Design a correct implementation of this function in MIPS assembly language; that is, hand-compile the C code above to equivalent MIPS assembly. You may use any standard MIPS pseudo instructions. Comment your code.

B.3 Using only the XOR, AND, OR and SLL instructions, write shortest possible MIPS program to add two unsigned numbers each of which is either zero or one (i.e., the sum will be zero, one or two). Assume that the source values are stored in $r2 and $r3, and that the result is stored in $r4.