9. The Memory Hierarchy (3)
Main Memory

Main memory is the name given to the level below the cache(s) in the memory hierarchy. There is a large variety of dimensions, but a smaller one in speed due to the fact that vendors use the same chips to build memory arrays. A main memory may have a few MBytes for a typical Personal Computer, tens to hundreds of MBytes for a workstation, hundreds of MBytes to GBytes for supercomputers. The capacity of main memory has continuously increased over the years, as prices have dramatically dropped. The main memory must satisfy the cache requests as quickly as possible (the main memory should have a low latency), and must provide sufficient bandwidth for I/O devices and for vector units (if it is the case).

The **access time**, is defined as the time between the moment the read command is issued and the moment the requested data is at outputs.
The **cycle time** is defined as the minimum time between successive accesses to memory. The cycle time is usually greater than the access time.

9.1 DRAM/SRAM

To access data in a memory chip with a capacity of NxM bits, one must provide a number of addresses equal to:

$\log_2 N$

N is the number of “words” each chip has; each “word” is M bits wide. As the technology improved, the packaging costs become a real concern, as the number of address lines got greater and greater.

Example 9.1 ADDRESS LINES:

Which is the number of address lines needed for a 4 Mbit memory chip:
a) organized as 4Mx1;
b) organized as 1Mx4?

Answer:
a) 4 M = 2^{22} hence the number of address lines is
$\log_2 2^{22} = 22$

b) 1 M = 2^{20}therefore the number of address lines is
$\log_2 2^{20} = 20$

To keep memory chips cheap, the solution adopted for Dynamic RAM (DRAM) integrated circuits was to multiplex the address, thus reducing the number of pins for addresses to half, and adding two new control lines: **RAS** (Row Address Strobe), which loads into an internal buffer half of the address supplied by the control on the address lines, and **CAS** (Column Address Strobe) which handles the second half of the address.

Example 9.2 NUMBER OF PINS:

How many pins has a 1Mx1 memory chip:
a) in DRAM technology;
b) in SRAM technology?

Answer:
Organization is important because it says how many pins are needed for data input and data output lines; it still does not say everything about the chip, more precisely it does not say if the input and output lines are the same or are separate.

Let's suppose that the chip in this example has one input line and one output line.

a) 1 M = 2^{20}
$n_A = \log_2 2^{20} = 20$ addresses

a)For DRAM the number of address lines (pins) is half of the address size:

10	address lines	
1	RAS	
1	CAS	
1	WE	(Write Enable)
1	Din	(the data input line)
1	Dout	(the data output line)
2	for power supply	

Total 17 pins needed for 1Mx1 DRAM. A real circuit has 18 pins, with one pin unused but devoted to use as an address line in the 4Mx1 chips.

b)For SRAM the number of address lines is the same as the address size:

20	address lines
1	WE
1	Din
1	Dout
2	for power supply

Total 25 pins needed for a 1Mx1 SRAM which fits in a 26 pin chip.

The reason for which SRAM address lines are not multiplexed is **speed;** the package is however more expensive than the package for a DRAM with the same capacity.

Another problem the designer faces, when using DRAM circuits is the refresh: each row in a DRAM circuit has to be accessed within some time interval (say 2 milliseconds), to prevent data from getting lost. This is a consequence of the dynamic technology, where data is stored as electric charge in a small capacitor: due to unavoidable losses in the dielectric of the capacitor, the charge decreases in time; the purpose of refresh is to periodically restore the charge on the capacitors (a charged capacitor stores a 1) thus preserving data.

The refresh requirement implies that the memory will be sometimes unavailable, being busy to do refresh: usually it takes a memory cycle to do a refresh to one row in DRAM. The number of refresh cycles in the critical time interval is a circuit specification, and can be found in the circuit's data sheet. Multiplexed address also mean an involved timing for DRAM memories; the cycle time is always larger than the access time.

SRAM use more transistors per memory-bit to prevent the loss of data. The cycle time for a static memory is very close to the access time, and is in the range of nanoseconds to tens of nanoseconds.

With the today's technology the maximum capacity of DRAMs is 16 times larger than that of SRAMs; the cycle time for a SRAM is, on the other hand, 8 to 16 times shorter than that of a DRAM.

As we saw in chapter 8, the faster a CPU is, the higher the miss penalty is. DRAM capacity has increased, over the last decade, quadrupling every three years. Unfortunately this is not the case with the DRAM performance: the 64 Kbit circuit, introduced in 1980, had a cycle time of 250 ns, while the 4 Mbit circuit introduced in 1989 has a cycle time of 165 ns; the capacity is 64 times larger but the performance is only 51% better (figures for the access time are quite similar).

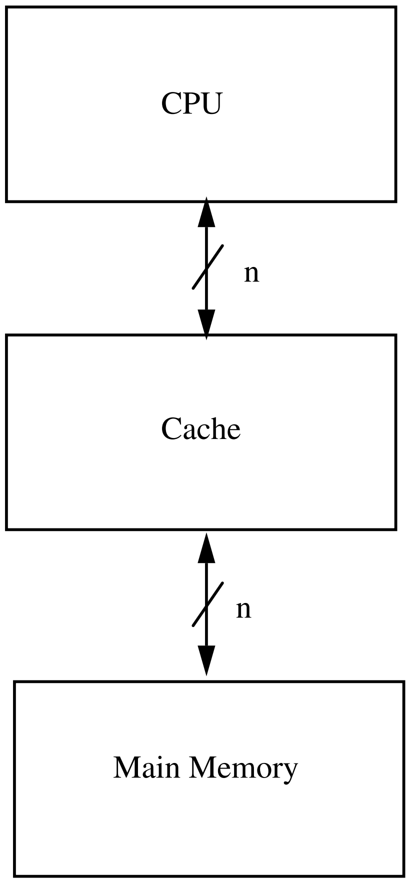


FIGURE 9.1 Connecting the Main memory to the CPU and cache; all buses have the same width.

9.2 Possible Organizations for Main Memory

We shall compare different memory organizations based on the following assumptions:

- all transfers are multiple of word (1 word = 4 bytes);
- 1 clock cycle to send the address;
- 10 clock cycles for the access time;
- 1 clock cycle for a bus transfer of the accessed item.

The simple and cheap approach for memory organization is to have transfers, between all levels of the memory hierarchy, the same size, as depicted in Figure 9.1.

Example 9.3 MEMORY ORGANIZATION:

Compute the miss penalty and the memory bandwidth for a word organized memory system. The cache block size is 8 words (32 bytes).

Answer:
For each word in the block the address must be transmitted (1 clock cycle), a fixed amount of time has to be spent waiting (10 clock cycles), and each word has to be transferred into the cache (1 clock cycle); therefore the miss penalty is:

miss_penalty = 8*(1 + 10 + 1) = 96 clock cycles

The memory bandwidth is:

$$\text{memory\_bandwidth} = \frac{\text{bytes\_transferred}}{\text{clock\_cycles}}$$

$$\text{memory\_bandwidth} = \frac{32}{96} = 0.33 \text{ bytes/clock cycle}$$

There are two parameters that can be modified to obtain a larger memory bandwidth: to increase the number of bytes transferred in the same amount of time, and to decrease the number of clock cycles necessary to complete a block transfer. The two possibilities correspond to two different memory organizations: a wider memory and interleaved memory respectively.

Wider Main Memory

The basic organization of a wider memory is presented in Figure 9.2. The data bus between the cache and main memory is wider than the bus between the cache and the CPU (the size of this one is the size of the

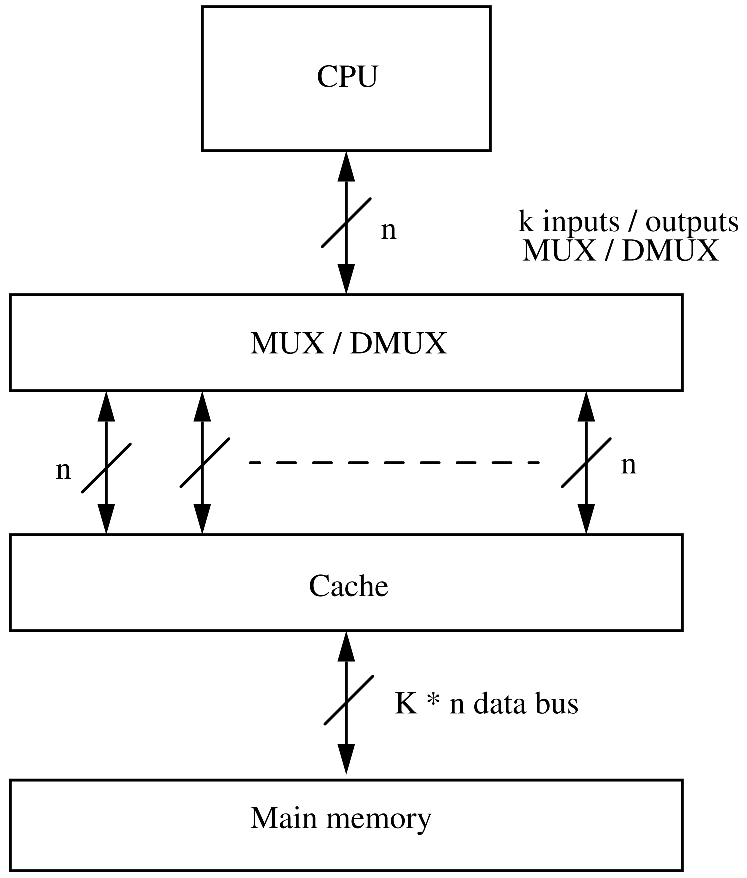


FIGURE 9.2 Wide memory organization. k * n bits are transferred between main memory and the cache, but only n bits between CPU and the cache.

datapath). In this case the address no longer indicates a word in main memory but a block; it is, in other words, the block-frame address that comes to main memory.

Example 9.4 MISS PENALTY AND MEMORY BANDWIDTH:

Calculate the miss penalty and the bus bandwidth for a wide memory organization. The cache line is 8 words wide, and the data bus is also 8 words wide.

Answer:
It takes one clock cycle to transmit the address, 10 clock cycles to access a line in the memory (8 words are accessed at once), and one more clock cycle to transfer the line into the cache; hence, the miss penalty is:

miss_penalty = 1 + 10 + 1 = 12 clock cycles

and the memory bandwidth is:

$$\text{memory\_bandwidth} = \frac{\text{bytes\_transferred}}{\text{clock\_cycles}}$$

$$\text{memory\_bandwidth} = \frac{32}{12} = 2.67 \text{ bytes/clock cycle}$$

The memory bandwidth is 8 times larger than for the word organized memory.

There is a price to be paid for better performance: because the CPU reads only n bits at a time and the cache is n*k bits organized, there must be a multiplexer between the cache and the CPU. Incidentally this is the case with cache represented in Figure 8.5 where we have a multiplexer that selects the proper word from the block (note also that, if the memory is byte addressable the scheme gets more involved, in that the type of item being addressed must be stated).

Another problem with the wide memories is related to the price paid by the customer: if the chips available for expansion have a NxM capacity, then the number of circuits the user must add into the system is a multiple of:

n*k / M

because the user has to add complete lines to the memory for expansion.

Interleaved Memory

The memory can be organized in banks, each bank one word wide, such that transfers between the cache and main memory are word wide, but several words can be read at once and then transferred one after the other to the cache. Figure 9.3 presents a interleaved memory organization.

Having an interleaved memory there is only one address that the CPU has to supply to main memory: this address will access a word in the bank number:

(address) modulo (number of banks)

while the other banks will access words at addresses successive to the address issued by the CPU.

Example 9.5 MISS PENALTY AND MEMORY BANDWIDTH:

Compute the miss penalty and the memory bandwidth for a 4 bank main memory; each bank is one word wide. The cache is 8 words wide.

Answer:
Because the memory has 4 banks, there will be 4 words read in a single burst; this takes one clock cycle to send the address, 10 clock cycles waiting time for the memories to access date, and 4 clock cycles to read the four word coming from the four banks. Since the cache block is 8 words wide this process has to be repeated; therefore, the miss penalty is:

miss_penalty = 2*(1 + 10 + 4*1) = 30 clock cycles

and the memory bandwidth is:

$$\text{memory\_bandwidth} = \frac{\text{bytes\_transferred}}{\text{clock\_cycles}}$$

$$\text{memory\_bandwidth} = \frac{32}{30} = 1.1 \text{ bytes/clock cycle (roughly)}$$

Interleaving gives fairly good performance as compared with a word organized main memory, still having the advantage that it does not require a wider data bus. The manner in which addresses are mapped to banks affect the memory's behavior; mapping word addresses to banks, with banks word wide, is a natural solution for the today's 32 bit machines that access the most frequent words in the memory.

An interleaved memory behaves fine in the case of cache misses due to the fact that words are read sequentially, and transferred one at a time to the cache; it is also attractive for write-back caches, as the words in a block are written sequentially into the memory, with the price of a single access time.

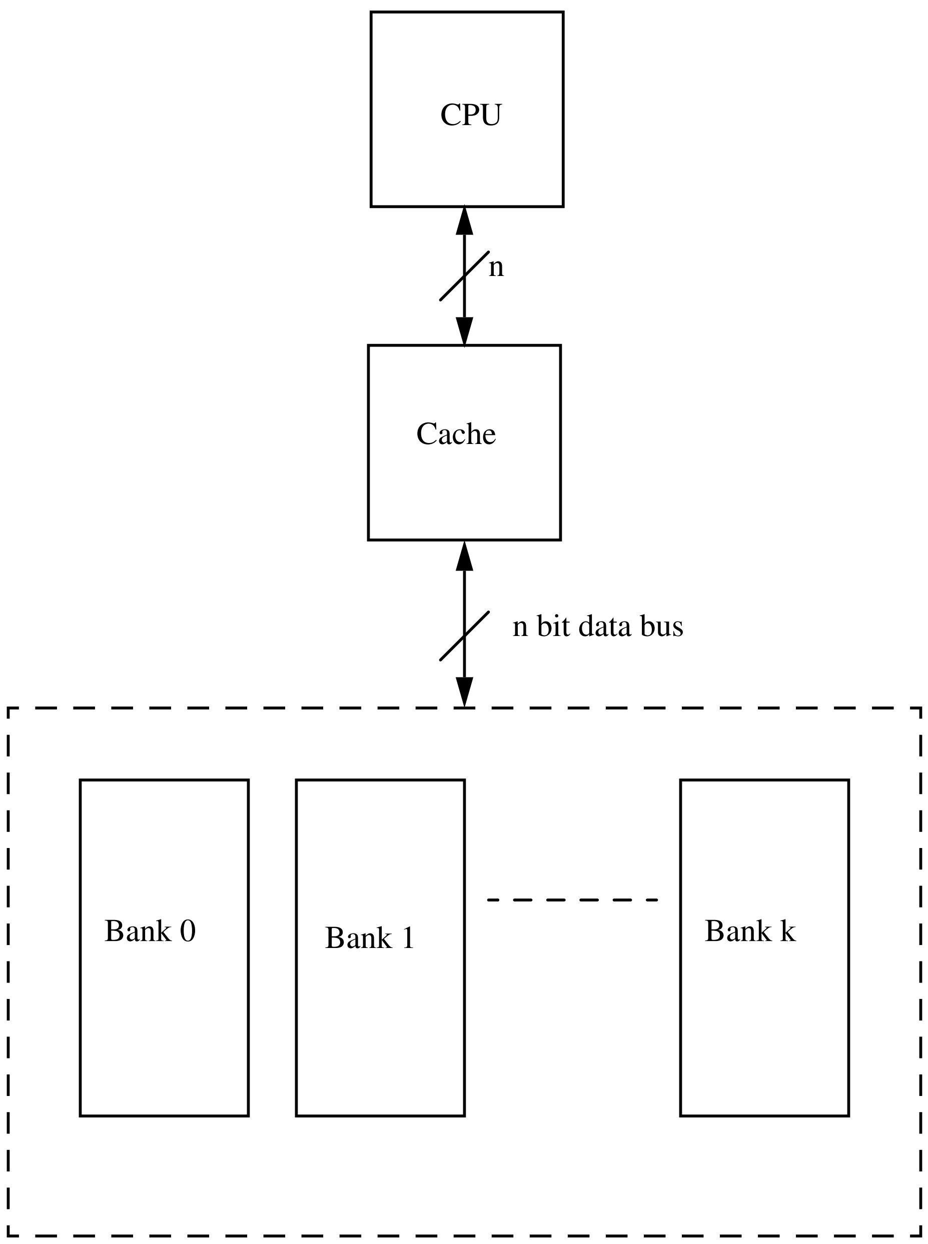


FIGURE 9.3 Interlaced memory organization. CPU, cache and each memory bank have the same width.

There is however the same drawback as for the wide memory organization, the price that the user must pay to increase the system's memory capacity. In this case the user must add the same number of chips in each bank, for a memory upgrade. Moreover, as the memory chips get larger capacities, it is more difficult to organize the chips in banks, as the following example suggests.

Example 9.6 MEMORY CAPACITY AND NUMBER OF CHIPS:

The maximum memory a PC can address is 16 MBytes; you have to design a four bank interleaved memory for this system. Each bank is byte organized. How many chips are necessary, using:
a) 1Mx1 chips;
b) 4Mx1 chips;
c) 16Mx1 chips?

Answer:
The capacity of a bank is:

$$\text{bank\_capacity} = \frac{\text{memory\_capacity}}{\text{number\_of\_banks}}$$

$$\text{bank\_capacity} = \frac{16\text{MB}}{4} = 4\text{MB/bank}$$

a)

$$n_a = (4 \text{ banks}) * \left(\frac{4\text{Mx}8}{1\text{Mx}1}\text{circuits per bank}\right) = 4 * 32 = 128 \text{ circuits}$$

b)

$$n_b = (4 \text{ banks}) * \left(\frac{4\text{Mx}8}{4\text{Mx}1}\text{circuits per bank}\right) = 4 * 8 = 32 \text{ circuits}$$

c)

$$n_a = (4 \text{ banks}) * \left(\frac{4\text{Mx}8}{16\text{Mx}1}\text{circuits per bank}\right) = 4 * 2 = 8 \text{ circuits ??}$$

We are in trouble because we need at least 8 circuits per bank to ensure the proper number of inputs and outputs (using 16Mx1 circuits); using 8 such circuits per bank means a capacity of:

8 * 16Mx1 = 16 MByte/bank

If the system can access only 16 MB, it results that 3/4 of the main memory is inaccessible.

Given the actual conditions in which the CPU performance increases at a faster pace than the memory performance, memory organizations that reduce the cache penalty tend to become common place.

Exercises

9.1 A memory hierarchy is being designed for a system. The following possibilities have to be investigated:
a) cache block size is 1 word, miss rate is 20%
b) cache block size is 4 words, miss rate is 10%
c) cache block size is 8 words, miss rate is 2%
In every case there are 1.4 memory accesses per instruction. Which is the best choice for the main memory? State your assumptions.

9.2 Explore the possibility of using some of the features the new DRAM circuits offer, to improve the memory performance: here are some of the standard DRAM improvements you might want to consider:
a) nibble mode;
b) page mode;
c) static column.

9.3 A new idea being studied is to move the cache closer to the memory, more precisely on the same memory chip die; this is tempting because, in the case of read, a whole row is accessed: for a 1Mx1 DRAM memory a row is 1024 bits wide (supposing the die is square). How do you think could this improve the memory performance? For a good introduction in this, you could consider a series of articles in the IEEE Spectrum, October 1992.

Performance Metrics

Why study performance metrics?

- determine the benefit/lack of benefit of designs
- computer design is too complex to intuit performance & performance bottlenecks
- have to be careful about what you mean to measure & how you measure it

What you should get out of this discussion

- good metrics for measuring computer performance
- what they should be used for
- what metrics you shouldn’t use & how metrics are misused

Performance of Computer Systems

Many different factors to take into account when determining performance:

- Technology
    - circuit speed (clock, MHz)
    - processor technology (how many transistors on a chip)
- Organization
    - type of processor (ILP)
    - configuration of the memory hierarchy
    - type of I/O devices
    - number of processors in the system
- Software
    - quality of the compilers
    - organization & quality of OS, databases, etc.

“Principles” of Experimentation

Meaningful metrics

execution time & component metrics that explain it

Reproducibility

machine configuration, compiler & optimization level, OS, input

Real programs

no toys, kernels, synthetic programs

SPEC is the norm (integer, floating point, graphics, webserver)

TPC-B, TPC-C & TPC-D for database transactions

Simulation

long executions, **warm start** to mimic **steady-state** behavior

usually applications only; some OS simulation

simulator “validation” & internal checks for accuracy

Metrics that Measure Performance

Raw speed: peak performance (never attained)

Execution time: time to execute one program from beginning to end

- the “performance bottom line”
- wall clock time, response time
- Unix time function: 13.7u 23.6s 18:27 3%

Throughput: total amount of work completed in a given time

- transactions (database) or packets (web servers) / second
- an indication of how well hardware resources are being used
- good metrics for chip designers or managers of computer systems

(Often improving execution time will improve throughput & vice versa.)

Component metrics: subsystem performance, e.g., memory behavior

- help explain how execution time was obtained
- pinpoints performance bottlenecks

Execution Time

$\text{Performance}_a = 1 / (\text{Execution Time}_a)$

Processor A is faster than processor B, i.e.,

$\text{Execution Time}_A < \text{Execution Time}_B$

$\text{Performance}_A > \text{Performance}_B$

Relative Performance

$\text{Performance}_A / \text{Performance}_B$

$= n$

$= \text{ExecutionTIme}_B / \text{ExecutionTime}_A$

performance of A is *n* times greater than B

execution time of B is *n* times longer than A

CPU Execution Time

The time the CPU spends executing an application

- no memory effects
- no I/O
- no effects of multiprogramming

CPUExecutionTime = CPUClockCycles * ClockCycleTime

Cycle time (clock period) is measured in time or rate

- clock cycle time = 1/clock cycle rate

  CPUExecutionTime = CPUClockCycles / ClockCycleRate

- clock cycle rate of 1 MHz = cycle time of 1 μs
- clock cycle rate of 1 GHz = cycle time of 1 ns

CPI

CPUClockCycles = NumberOfInstructions * CPI

Average number of clock cycles per instruction

- throughput metric
- component metric, not a measure of performance
- used for processor organization studies, given a fixed compiler & ISA

Can have different CPIs for classes of instructions
e.g., floating point instructions take longer than integer instructions

$$CPUClockCycles = \sum_{1}^{n} (CPI_i \times C_i)$$

where CPI_i = CPI for a particular class of instructions

where C_i = the number of instructions of the ith class that have been executed

Improving part of the architecture can improve a CPI_i

- Talk about the contribution to CPI of a class of instructions

CPU Execution Time

CPUExecutionTime =
numberofInstructions * CPI * clockCycleTime

To measure:

- execution time: depends on all 3 factors
    - time the program
- number of instructions: determined by the ISA
    - programmable hardware counters
    - profiling
        - count number of times each basic block is executed
        - instruction sampling
- CPI: determined by the ISA & implementation
    - simulator: interpret (in software) every instruction & calculate the number of cycles it takes to simulate it
- clock cycle time: determined by the implementation & process technology

Factors are interdependent:

- RISC: increases instructions/program, but decreases CPI & clock cycle time because the instructions are simple
- CISC: decreases instructions/program, but increases CPI & clock cycle time because many instructions are more complex

Metrics Not to Use

MIPS (millions of instructions per second)

instruction count / execution time*10^6 =
clock rate / (CPI * 10^6)

- instruction set-dependent (even true for similar architectures)
- implementation-dependent
- compiler technology-dependent
- program-dependent

+ intuitive: the higher, the better

MFLOPS (millions of floating point operations per second)

floating point operations / (execution time * 10^6)

+ FP operations are independent of FP instruction implementation

- different machines implement different FP operations
- different FP operations take different amounts of time
- only measures FP code

static metrics (code size)

Means

Measuring the performance of a workload

- **arithmetic**: used for averaging execution times

$$\left(\sum_{i=1}^{n} time_i\right)\times\frac{1}{n}$$

- **harmonic**: used for averaging rates ("the average of", as opposed to "the average statistic of")

$$\frac{p}{\left(\sum_{i=1}^{p} 1/rate_i\right)}$$

- weighted means: the programs are executed with different frequencies, for example:

$$\left(\sum_{i=1}^{n} time_i \times weight_i\right)\times\frac{1}{n}$$

Means

	FP Ops	Time (secs)		
		Computer A	Computer B	Computer C
program 1	100	1	10	20
program 2	100	1000	100	20
total		1001	110	40
arith mean		500.5	55	20

	FP Ops	Rate (FLOPS)		
		Computer A	Computer B	Computer C
program 1	100	100	10	5
program 2	100	.1	1	5
harm mean		.2	1.5	5
arith mean		50.1	5.5	5

Computer C is ~25 times faster than A when measuring execution time

Still true when measuring MFLOPS(a rate) with the harmonic mean

Speedup

Speedup = $\text{Execution Time}_{\text{beforeImprovement}} / \text{ExecutionTime}_{\text{afterImprovement}}$

Amdahl's Law:

Performance improvement from speeding up a part of a computer system is limited by the proportion of time the enhancement is used.

VIT® UNIVERSITY (Estd. u/s 3 of UGC Act 1956) Vellore-632 014, Tamil Nadu, India.

Problem Set

1. Perform the division of (117) ÷ (-9) using Restoring and Non restoring division algorithm.

2. Show the contents of registers E, A, Q and SC during the process of multiplication of two binary numbers, 11111 (multiplicand) and 10101 (multiplier) using Booths Algorithm.

3. Show step by step process of modified booth multiplication algorithm with the flowchart for 15 x -13.

4. Give examples for the following four different cases (i) carry, overflow (ii) no carry, overflow (iii) carry, no overflow (iv)no carry, no overflow.

5. Give the algorithm for Non restoring division and perform the same on (27) ÷ (-6)

6. How do you determine the overflow in each of the representations? (i) signed magnitude addition and subtraction (ii) two's complement addition and subtraction (iii) unsigned addition and subtraction? Illustrate each case with an example.

7. Give the algorithm for 2's complement addition and subtraction and perform the same on

1011111 + 0101101.

8. Compare and contrast Booth with modified booth multiplication Algorithm.

9. Represent -35 and 75 in a) Sign magnitude b) ones complement c) twos complement format.

10. State and prove the Bit pair recording table of modified Booths Algorithm.

11. Represent +12.456 in single precision IEEE 754 format.

12. Represent – 23.543 in double precision IEEE 754 format.

13. Determine the value of 1 10101011 11100101000000000000000 in decimal.

14. Determine the value of

0 11101011010 1110101010000000000000000000000000000000000000000

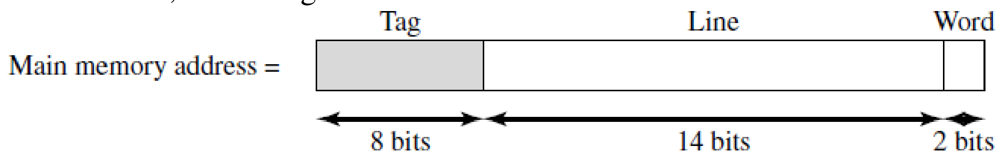
15. For the hexadecimal main memory addresses 111111, 666666,BBBBBB, show the following information, in hexadecimal format:

a. Tag, Line, and Word values for a direct-mapped cache, where tag – 8bits, line – 14 bits, word – 2 bits
b. Tag and Word values for an associative cache, where tag – 22 bits, word – 2 bits

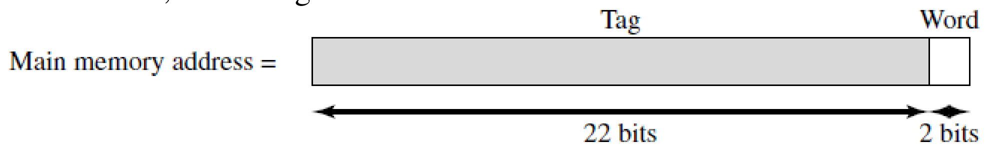
c. Tag, Set, and Word values for a two-way set-associative cache, where tag – 9 bits, set – 13 bits, word – 2 bits

16. List the following values:

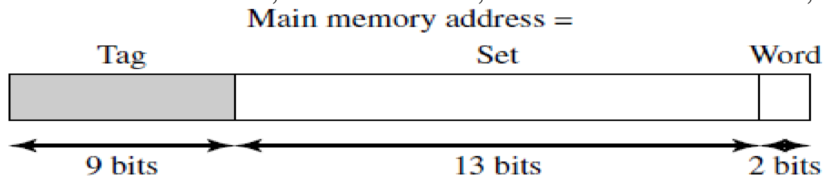
d. For the direct cache from the below Fig: address length, number of addressable units, block size, number of blocks in main memory, number of lines in cache, size of tag



e. For the associative cache from below Figure: address length, number of addressable units, block size, number of blocks in main memory, number of lines in cache, size of tag



f. For the two-way set-associative cache example of Figure 4.15: address length, number of addressable units, block size, number of blocks in main memory, number of lines in set, number of sets, number of lines in cache, size of tag



17. Consider a 32-bit microprocessor that has an on-chip 16-KByte four-way set-associative cache. Assume that the cache has a line size of four 32-bit words. Draw a block diagram of this cache showing its organization and how the different address fields are used to determine a cache hit/miss. Where in the cache is the word from memory location ABCDE8F8 mapped?

18. Consider a machine with a byte addressable main memory of 216 bytes and block size of 8 bytes. Assume that a direct mapped cache consisting of 32 lines is used with this machine.

g. How is a 16-bit memory address divided into tag, line number, and byte number?

h. Into what line would bytes with each of the following addresses be stored?

0001 0001 0001 1011
1100 0011 0011 0100
1101 0000 0001 1101
1010 1010 1010 1010

i. Suppose the byte with address 0001 1010 0001 1010 is stored in the cache. What are the addresses of the other bytes stored along with it?

j. How many total bytes of memory can be stored in the cache?

k. Why the tag is also stored in the cache?

19. A set-associative cache has a block size of four 16-bit words and a set size of 2. The cache can accommodate a total of 4096 words. The main memory size that is cacheable is 64K × 32 bits. Show how the processor's addresses are interpreted?

20. Consider a memory system that uses a 32-bit address to address at the byte level, plus a cache that uses a 64-byte line size.

l. Assume a direct mapped cache with a tag field in the address of 20 bits. Show the address format and determine the following parameters: number of addressable units, number of blocks in main memory, number of lines in cache, size of tag.
m. Assume an associative cache. Show the address format and determine the following parameters: number of addressable units, number of blocks in main memory, number of lines in cache, size of tag.
n. Assume a four-way set-associative cache with a tag field in the address of 9 bits. Show the address format and determine the following parameters: number of addressable units, number of blocks in main memory, number of lines in set, number of sets in cache, number of lines in cache, size of tag.

21. Consider a computer with the following characteristics: total of 1Mbyte of main memory; word size of 1 byte; block size of 16 bytes; and cache size of 64 Kbytes.

o. For the main memory addresses of F0010, 01234, and CABBE, give the corresponding tag, cache line address, and word offsets for a direct-mapped cache.
p. Give any two main memory addresses with different tags that map to the same cache slot for a direct-mapped cache.
q. For the main memory addresses of F0010 and CABBE, give the corresponding tag and offset values for a fully-associative cache.
r. For the main memory addresses of F0010 and CABBE, give the corresponding tag, cache set, and offset values for a two-way set-associative cache.

22. Consider the following code:

```
for (i = 0; i < 20; i++)
        for ( j = 0; j < 10; j++)
                a[i] =  a[i] * j
```

a. Give one example of the spatial locality in the code.
b. Give one example of the temporal locality in the code.

23. Consider a memory system with the following parameters:
$T_c = 100ns$; $T_m = 1200ns$
If the effective access time is 10% greater than the cache access time, what is the hit ratio H for look through cache?

24. Consider a look through cache with an access time of 1 ns and a hit ratio of H 0.95. Suppose that we can change the cache design (size of cache, cache organization) such that we increase H to 0.97, but increase access time to 1.5 ns. What conditions must be met for this change to result in improved performance?

25. A computer employs RAM chips of 128 x 8 and ROM chips of 512 x 8. The computer system needs 256 *16 of RAM, 1024 x 16 of ROM, and two interface units with 256 registers each.

Show the chip layout for the given specifications.

26. A computer employs RAM chips of 128 x 8 and ROM chips of 512 x 8. The computer system needs 512 *8 of RAM, 512 x 16 of ROM, and two interface units with 256 registers each.
Show the chip layout for the given specifications.

27. If the received data is 101110011001. Determine whether single bit error or more than one bit error occurs. If there is an error in single bit, perform error correction.

28. If the received data is 001101100111. Determine whether single bit error or more than one bit error occurs. If there is an error in single bit, perform error correction.

29. Calculate the check bits for the data bits 101100110101.

30. Compare the performance of FIFO, LRU and optimal page replacement algorithms in terms of hit ratio considering three pages to be in memory at a time for the following sequence.
2 3 2 1 5 2 4 5 3 2 5 2

ANSWER KEY

Name of Examination	**Final Assessment Test (FAT), Fall 2019-20 Semester, (NOV 2019)**			
Slot: G2	Course Mode: **CBL / PBL / RBL**	**Class Number (s):** 0616		
Course Code:	CSE2001	Course Title:	COMPUTER ARCHITECTURE AND ORGANIZATION	
Emp. ID :	16386	Faculty Name:	D.RUBY	School: SCOPE
Contact No.	9043542311	Email:	ruby.d@vit.ac.in	

1. The processor A, B C and D has a 2 GHz clock frequency. Find the total execution time for the programme with instruction mix given below. If the CPI of arithmetic instruction was doubled what would be the impact on the execution time of all the processors.

Processors	Instruction mix / Processor			CPI		
	Arithmetic	Load / Store	Branch	Arithmetic	Load / Store	Branch
A	2560	1280	256	1	4	2
B	1280	640	128	1	4	2
C	640	320	64	1	4	2
D	320	160	32	1	4	2

Find the total execution time for this program on A, B, C & D processors. Assume that each processor has a 2 GHz clock frequency. If the CPI of arithmetic instructions was doubled, what would the impact be on the execution time of the program on A, B, C & D processors?

ANS: i) Total execution time:
A = 4.096 μs, B = 2.048 μs, C = 1.024 μs, D = 0.512 μs
ii) Execution time after CPI been doubled
A = 5.376 μs, B = 2.688 μs, C = 1.344 μs, D = 0.672 μs

2. Calculate M multiplied by Q using Booth's Algorithm where M = -11 and Q = 27.

ANS:

n	A	Q	Q_0	Comments
6	000000	011011	0	Initial Values
5	001011 000101	011011 101101	0 1	A<- A-M Arithmetic Shift Right

4	000010	110110	1	Arithmetic Shift Right
3	110111 111011	110110 111011	1 0	A<-A+M Arithmetic Shift Right
2	000110 000011	111011 011101	0 1	A<- A-M Arithmetic Shift Right
1	000001	101110	1	Arithmetic Shift Right
0	110110 111011	101110 010111	1 0	A<-A+M Arithmetic Shift Right

3. Write a program to evaluate the expression X= (A+B)*(C/D) with one address, two address and three address Instructions.

ANS:

ONE ADDRESS	TWO ADDRESS	THREE ADDRESS
LOAD A	MOV T, A	ADD X, A, B
ADD B	ADD T, B	DIV T, C, D
STORE T	MOV S, C	MUL X, X, T
LOAD C	DIV S, D	
DIV D	MUL T, S	
MUL T	MOV X, T	
STORE X		

4. A. Give a block diagram for a 8M X 32 memory using 512K X 8 RAM chips.

ANS:
- 16 rows (of four 512×8 chips) are needed.
- Address lines A18−0 are connected to all chips.
- Address lines A22−19 are connected to a 4-bit decoder to select one of the 16 rows.

B. Multiply the numbers $(0.5)_{10}$ and $(-0.4375)_{10}$ using binary floating point multiplication.

ANS: $(1.000 \times 2^{-1}) \times (-1.110 \times 2^{-2})$
$= -1.110 \times 2^{-3}$

5. The following is a list of 32-bit memory address references, given as word addresses of 8-bit each. 1, 134, 212, 1, 135, 213, 162, 161, 2, 44, 41, 221. For the above references, identify the binary address and the index address given a direct-mapped cache with initially 2-word blocks and a total size of blocks. Assuming the cache to be empty initially, list the hit or miss for cache references.

ANS: i) Binary Address: 00000001, 10000110, 11010100, 00000001, 10000111, 11010101, 10100010, 10100001, 00000010, 00101100, 00101001, 11011101

ii) Index: Binary address mod 16

iii) Hit/Miss: M, M, M, H, M, M, M, M, M, M, M, M.

6. Consider a two level memory hierarchy of the form (L1, L2) where L1 is connected directly to the CPU. Determine the average cost per bit and average access time for the data given below.

Level	Capacity	Cost	Access time	Hit ratio
L1	1024	0.1000	10^{-8}	0.9000
L2	2^{16}	0.0100	10^{-6}	-

ANS:

Average Cost (C) = $(C_1S_1 + C_2S_2) / (S_1 + S_2)$

$= ((0.1 \text{ X } 1024) + (0.01 \text{ X } 2^{16})) / (1024 + 2^{16})$

$= 0.011384615$

Average access time

$T_A = hT_{A1} + (1 - h) T_{A2}$

$= 0.9000 \text{ X } 10^{-8} + (1 - 0.9000) \text{ X } 10^{-6}$

$= 1.09 \text{ X } 10^{-7}$

7. It is necessary to transfer 512 words from a backup store to a memory section starting from address 1000 and the transfer is by means of DMA. i) What are the initial values that the CPU must transfer to the DMA controller. ii) Give step by step account of the actions taken during the input of the first two words.

ANS:

i) CPU initiates DMA by Transferring: 512 to the word count register. 1000 to the DMA address register. Bits to the control register to specify a write operation.

ii) 1. I/O device sends a “DMA request”.
2. DMA sends BR (bus request) to CPU.
3. CPU responds with a BG (bus grant).
4. Contents of DMA address register are placed in address bus.
5. DMA sends “DMA acknowledge” to I/O device and enables the write control line to memory.
6. Data word is placed on data bus by I/O device.
7. Increment DMA address register by 1 and Decrement DMA word count register by 1.

8. Repeat steps 4-7 for each data word Transferred.

8. List and explain the levels of RAID. What is the distinction between parallel access and independent access in context of RAID?

   ANS:
   RAID Level 0: Non-Redundant
   RAID Level 1: Mirrored
   RAID Level 2: Redundancy through hamming code
   RAID Level 3: Bit-interleaved parity
   RAID Level 4: Block level parity
   RAID Level 5: Block-level distributed parity

9. Assume a pipeline with four stages: fetch instruction (FI), decode instruction and calculate addresses (DA), fetch operand (FO), and execute (EX). Draw a diagram for a sequence of 7 instructions, in which the third instruction is a branch that is taken and in which there are no data dependencies.

   ANS:

	1	2	3	4	5	6	7	8	9	10
I1	FI	DA	FO	EX						
I2		FI	DA	FO	EX					
I3			FI	DA	FO	EX				
I4				FI	DA	FO				
I5					FI	DA				
I6						FI				
I7							FI	DA	FO	EX

10. A. Discuss the difference between tightly coupled multiprocessors and loosely coupled multiprocessors from the viewpoint of hardware organization and programming techniques.

   ANS:

   Tightly coupled multiprocessors require that all processed in the system have access to a common global memory. In loosely coupled multiprocessors, the memory is distributed and a mechanism is required to provide message-passing between the processors. Tightly coupled systems are easier to program since no special steps are required to make shared data available to two or more processors. A loosely coupled system required that sharing of data be implemented by the messages.

   B. What is the use of parity bits in an error correction code? How many check bits are needed if the Hamming error correction code is used to detect single bit errors in a 2048-bit data word?

ANS: Need K check bits such that $2048 + K \leq 2^K - 1$.
The minimum value of K that satisfies this condition is 12.