4.18) Consider a cache of 4 lines of 16 bytes each. Main memory is divided into blocks of 16 bytes each. That is, block 0 has bytes with addresses 0 through 15, and so on. Now consider a program that accesses memory in the following sequence of addresses: Once: 63 through 70 Loop ten times: 15 through 21; 80 through 95

a. Suppose the cache is organized as direct mapped. Memory blocks 0, 4 etc. are assigned to line 1; blocks 1, 5, etc. to line 2; and so on. Compute the hit ratio.

b. Suppose the cache is organized as 2-way set associative, with 2 sets of 2 lines each. Even-numbered blocks are assigned to set 0 and odd-numbered blocks are assigned to set 1. Compute the hit ratio for the two-way set associative cache using the Least Recently Used replacement scheme.

a.

Access 63	1 Miss	Block 3 → Slot 3	
Access 64	1 Miss	Block 4 → Slot 0	
Access 65-70	6 Hits		
Access 15	1 Miss	Block 0 → Slot 0	First Loop
Access 16	1 Miss	Block 1 → Slot 1	
Access 17-31	15 Hits		
Access 32	1 Miss	Block 2 → Slot 2	
Access 80	1 Miss	Block 5 → Slot 1	
Access 81-95	15 Hits		
Access 15	1 Hit		Second Loop
Access 16	1 Miss	Block 1 → Slot 1	
Access 17-31	15 hits		
Access 32	1 Hit		
Access 80	1 Miss	Block 5 → Slot 1	
Access 81-95	15 hits		
Access 15	1 Hit		Third Loop
Access 16	1 Miss	Block 1 → Slot 1	
Access 17-31	15 hits		
Access 32	1 Hit		
Access 80	1 Miss	Block 5 → Slot 1	
Access 81-95	15 hits		
Access 15	1 Hit		Fourth Loop

… Pattern continues to the Tenth Loop

For lines 63-70	2 Misses	6 Hits
First loop 15-32, 80-95	4 Misses	30 Hits
Second loop 15-32, 80-95	2 Misses	32 Hits
Third loop 15-32, 80-95	2 Misses	32 Hits
Fourth loop 15-32, 80-95	2 Misses	32 Hits
Fifth loop 15-32, 80-95	2 Misses	32 Hits

Sixth loop 15-32, 80-95	2 Misses	32 Hits
Seventh loop 15-32, 80-95	2 Misses	32 Hits
Eighth loop 15-32, 80-95	2 Misses	32 Hits
Ninth loop 15-32, 80-95	2 Misses	32 Hits
Tenth loop 15-32, 80-95	2 Misses	32 Hits
Total:	24 Misses	324 Hits

Hit Ratio = 324/348 = 0.931

b.

Access 63	1 Miss	Block 3 $\rightarrow$ Set 1 Slot 2	
Access 64	1 Miss	Block 4 $\rightarrow$ Set 0 Slot 0	
Access 65-70	6 Hits		
Access 15	1 Miss	Block 0 $\rightarrow$ Set 0 Slot 1	First Loop
Access 16	1 Miss	Block 1 $\rightarrow$ Set 1 Slot 3	
Access 17-31	15 Hits		
Access 32	1 Miss	Block 2 $\rightarrow$ Set 0 Slot 0	
Access 80	1 Miss	Block 5 $\rightarrow$ Set 1 Slot 2	
Access 81-95	15 Hits		
Access 15	1 Hit		Second Loop
Access 16-31	16 Hits		
Access 32	1 Hit		
Access 80-95	16 Hits		

… All hits for the next eight iterations

For lines 63-70	2 Misses	6 Hits
First loop 15-32, 80-95	4 Misses	30 Hits
Second loop 15-32, 80-95	0 Misses	34 Hits
Third loop 15-32, 80-95	0 Misses	34 Hits
Fourth loop 15-32, 80-95	0 Misses	34 Hits
Fifth loop 15-32, 80-95	0 Misses	34 Hits
Sixth loop 15-32, 80-95	0 Misses	34 Hits
Seventh loop 15-32, 80-95	0 Misses	34 Hits
Eighth loop 15-32, 80-95	0 Misses	34 Hits
Ninth loop 15-32, 80-95	0 Misses	34 Hits
Tenth loop 15-32, 80-95	0 Misses	34 Hits

Total = 6 Misses 342 Hits
Hit Ratio = 342/348 = 0.983

4.19) Consider a memory system with the following parameters:

$T_c=100\text{ns}$ $C_c=10^{-4}\$/\text{bit}$
$T_m=1{,}200\text{ns}$ $C_m=10^{-5}\$/\text{bit}$

a. What is the cost of 1 MByte of main memory?

b. What is the cost of 1 MByte of main memory using cache memory technology?

c. If the effective access time is 10% greater than the cache access time, what is the hit ratio *H*?

a. Cost = $C_m \times 8 \times 10^6 = 8 \times 10^3$ ¢ = \$80

b. Cost = $C_c \times 8 \times 10^6 = 8 \times 10^4$ ¢ = \$800

c. From Equation (4.1) : $1.1 \times T_1 = T_1 + (1 - H)T_2$

$$(0.1)(100) = (1 - H)(1200)$$

$$H = 1190/1200$$

4.26) The performance of a single-level cache system for a read operation can be characterized by the following equation:

$$T_a = T_c + (1-H)\, T_m$$

Where T_a is the average access time, T_c is the cache access time, T_m is the memory access time (memory to processor register), and *H* is the hit ratio. For simplicity, we assume that the word in question is loaded into the cache in parallel with the load to processor register. This is the same form as the Equation (4.1).

a. Define T_b = time to transfer a line between cache and main memory, and *W*= fraction of write references. Revise the preceding equation to account for writes as well as reads, using a write-through policy.

b. Define W_b as the probability that a line in the cache has been altered. Provide an equation for T_a for the write-back policy.

a. $T_a = T_c + (1 - H)T_b + W(T_m - T_c)$

b. $T_a = T_c + (1 - H)T_b + W_b(1 - H)T_b = T_c + (1 - H)(1 + W_b)T_b$

4.29 The average miss penalty equals the miss penalty times the miss rate. For a line size of one word, average miss penalty = 0.032 x 5 = 0.16 clock cycles. For a line size of 4 words and the nonburst transfer, average miss penalty = 0.011 x 20 = 0.22 clock cycles. For a line size of 4 words and the burst transfer, average miss penalty = 0.011 x 8 = 0.132 clock cycles.

5.3) Figure 5.15 shows a simplified timing diagram for a DRAM read operation over a bus. The access time is considered to last from t_1 to t_2. Then there is a recharge time, lasting from t_2 to t_3, during which the DRAM chips will have to recharge before the processor can access them again.

a. Assume that the access time is 60 ns and the recharge time is 40 ns. What is the memory cycle time? What is the maximum data rate this DRAM can sustain, assuming a 1-bit output?

b. Constructing a 32-bit wide memory system using these chips yields what data transfer rate?

- **a.** Memory cycle time = 60 + 40 = 100 ns. The maximum data rate is 1 bit every 100 ns, which is 10 Mbps.
- **b.** 320 Mbps = 40 MB/s.

5.5) a. The length of a clock cycle is 100 ns. Mark the beginning of T_1 as time 0.Address Enable returns to a low at 75. $\overline{\text{RAS}}$ goes active 50 ns later, or time 125. Data must become available by the DRAMs at time 300 – 60 = 240. Hence, access time must be no more than 240 – 125 = 115 ns.

b. A single wait state will increase the access time requirement to 115 + 100 = 215 ns. This can easily be met by DRAMs with access times of 150 ns. Source: [PROT88].

5.6) The memory of a particular microcomputer is built from 64K x 1DRAMs. According to the data sheet, the cell array of the DRAM is organized into 256 rows. Each row must be refreshed at least once every 4 ms. Suppose we refresh the memory on a strictly periodic basis.

a. What is the time period between successive refresh requests?

b. How long a refresh address counter do we need?

a. The refresh period from row to row must be no greater than 4000/256 = 15.625 µs.

b. An 8-bit counter is needed to count 256 rows ($2^8 = 256$). Source: [PROT88].

5.7) Figure 5.16 shows one of the early SRAMs, the 16 x 4 Signetics 7489 chip, which stores 16 4-bit words.

a. List the mode of operation of the chip for each ~~CS~~ input pulse shown in Figure 5.16c.

b. List the memory contents of word locations 0 through 6 after pulse n.

c. What is the state of the output data leads for the input pulses h through m?

a.

pulse a = write	pulse f = write	pulse k = read
pulse b = write	pulse g = store-disable outputs	pulse l = read
pulse c = write	pulse h = read	pulse m = read
pulse d = write	pulse i = read	pulse n = store-disable outputs
pulse e= write	pulse j = read	

b. Data is read in via pins (D3, D2, D1, D0)
word 0 = 1111 (written into location 0 during pulse a)
word 1 = 1110 (written into location 0 during pulse b)
word 2 = 1101 (written into location 0 during pulse c)
word 3 = 1100 (written into location 0 during pulse d)
word 4 = 1011 (written into location 0 during pulse e)
word 5 = 1010 (written into location 0 during pulse f)
word 6 = random (did not write into this location 0)

c. Output leads are (O3, O2, O1, O0)
pulse h: 1111 (read location 0)
pulse i: 1110 (read location 1)
pulse j: 1101 (read location 2)
pulse k: 1100 (read location 3)
pulse l: 1011 (read location 4)
pulse m: 1010 (read location 5)

7.10 a. The device generates 8000 interrupts per second or a rate of one every 125 μs. If each interrupt consumes 100 μs, then the fraction of processor time consumed is 100/125 = 0.8

b. In this case, the time interval between interrupts is 16 × 125 = 2000 μs. Each interrupt now requires 100 μs for the first character plus the time for transferring each remaining character, which adds up to 8 × 15 = 120 μs, for a total of 220 μs. The fraction of processor time consumed is 220/2000 = 0.11

c. The time per byte has been reduced by 6 μs, so the total time reduction is 16 × 6 = 96 μs. The fraction of processor time consumed is therefore (220 – 96)/2000 = 0.062. This is an improvement of almost a factor of 2 over the result from part (b). Source: [PROT88].

7.13 a. For the actual transfer, the time needed is (128 bytes)/(50 KBps) = 2.56 ms. Added to this is the time to transfer bus control at the beginning and end of the transfer, which is 250 + 250 = 500 ns. This additional time is negligible, so that the transfer time can be considered as 2.56 ms.

b. The time to transfer one byte in cycle stealing mode is 250 + 500 + 250 = 1000 ns = 1 μs. Total amount of time the bus is occupied for the transfer is 128 μs. This is less than the result from part (a) by a factor of 20. Source: [PROT88].

7.17 Only one device at a time can be serviced on a selector channel. Thus,

$$\text{Maximum rate} = 800 + 800 + 2 \times 6.6 + 2 \times 1.2 + 10 \times 1 = 1625.6 \text{ KBytes/sec}$$

7.19 For each case, compute the fraction g of transmitted bits that are data bits. Then the maximum effective data rate ER is

$$ER = gR$$

a. There are 7 data bits, 1 start bit, 1.5 stop bits, and 1 parity bit.

$$g = 7/(1+7+1+1.5) = 7/10.5$$
$$ER = 0.67 \times R$$

b. Each frame contains 48 + 128 = 176 bits. The number of characters is 128/8 = 16, and the number of data bits is 16 × 7 = 112.

$$ER = (112/176) \times R = 0.64 \times R$$

c. Each frame contains 48 = 1024 bits. The number of characters is 1024/8 = 128, and the number of data bits is 128 × 7 = 896.

$$ER = (896/1072) \times R = 0.84 \times R$$

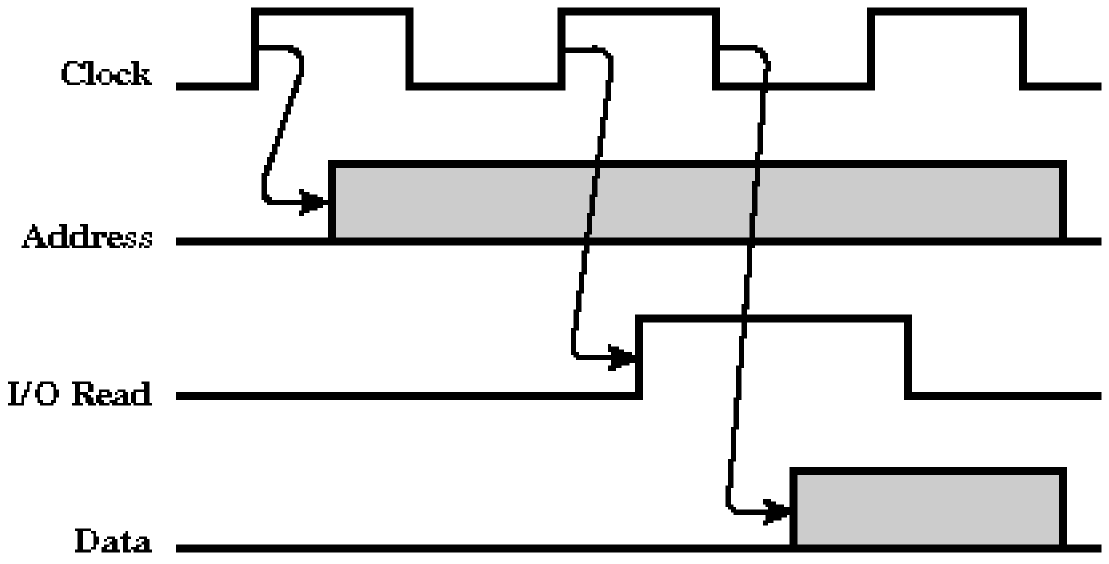
d. With 9 control characters and 16 information characters, each frame contains $(9 + 16) \times 8 = 200$ bits. The number of data bits is $16 \times 7 = 112$ bits.

$$ER = (112/200) \times R = 0.56 \times R$$

e. With 9 control characters and 128 information characters, each frame contains $(9 + 128) \times 8 = 1096$ bits. The number of data bits is $128 \times 7 = 896$ bits.

$ER = (896/1096) \times R = 0.82 \times R$

7.20 a. Assume that the women are working, or sleeping, or otherwise engaged. The first time the alarm goes off, it alerts both that it is time to work on apples. The next alarm signal causes apple-server to pick up an apple an throw it over the fence. The third alarm is a signal to Apple-eater that he can pick up and eat the apple. The transfer of apples is in strict synchronization with the alarm clock, which should be set to exactly match Apple-eater's needs. This procedure is analogous to standard synchronous transfer of data between a device and a computer. It can be compared to an I/O read operation on a typical bus-based system. The timing diagram is as follows:



On the first clock signal, the port address is output to the address bus. On the second signal, the I/O Read line is activated, causing the selected port to place its data on the data bus. On the third clock signal, the CPU reads the data.

A potential problem with synchronous I/O will occur if Apple-eater's needs change. If he must eat at a slower or faster rate than the clock rate, he will either have too many apples or too few.

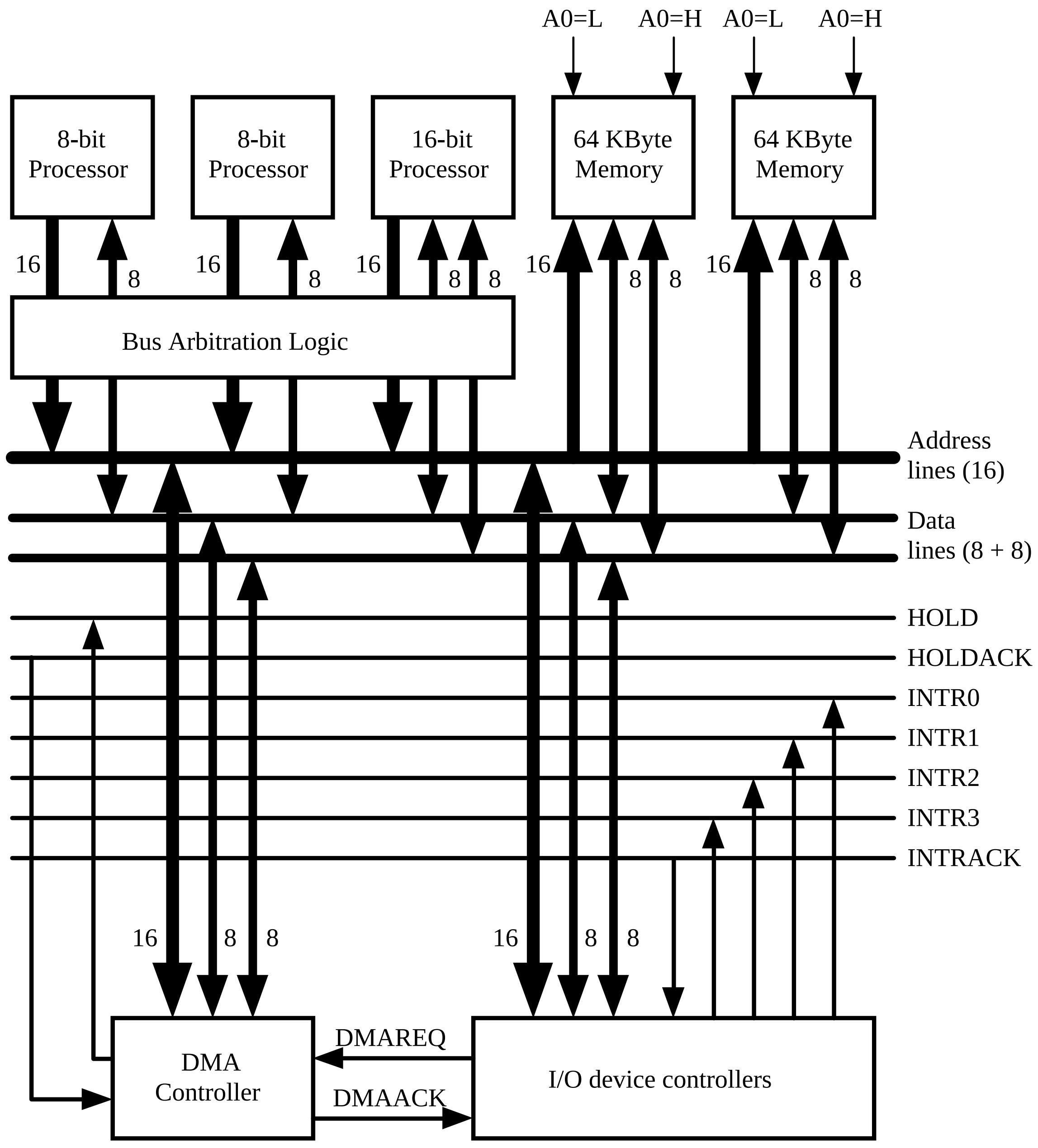
b. The women agree that Apple-server will pick and throw over an apple whenever he sees Apple-eater's flag waving. One problem with this approach is that if Apple-eater leaves his flag up, Apple-server will see it all the time and will inundate her friend with apples. This problem can be avoided by giving Apple-server a flag and providing for the following sequence:

1. Apple-eater raises her "hungry" flag when ready for an apple.
2. Apple-server sees the flag and tosses over an apple.
3. Apple-server briefly waves her "apple-sent" flag
4. Apple-eater sees the "apple-sent" flag, takes down her "hungry" flag, and grabs the apple.
5. Apple-eater keeps her "hungry" flag stays down until she needs another apple.

This procedure is analogous to asynchronous I/O. Unfortunately, Apple-server may be doing something other than watching for her friend's flag (like sleeping!). In that case, she will not see the flag, and Apple-eater will go hungry. One solution is to not permit apple-server to do anything but look for her friend's flag. This is a polling, or wait-loop, approach, which is clearly inefficient.

c. Assume that the string that goes over the fence and is tied to Apple-server's wrist. Apple-eater can pull the string when she needs an apple. When Apple-server feels a tug on the string, she stops what she is doing and throws over an apple. The string corresponds to an interrupt signal and allows Apple-server to use her time more efficiently. Moreover, if Apple-server is doing something really important, she can temporarily untie the string, disabling the interrupt.

7.21


A0=L
A0=H
A0=L
A0=H
8-bit Processor
8-bit Processor
16-bit Processor
64 KByte Memory
64 KByte Memory
16
8
16
8
16
8
8
16
8
8
16
8
8
Bus Arbitration Logic
Address lines (16)
Data lines (8 + 8)
HOLD
HOLDACK
INTR0
INTR1
INTR2
INTR3
INTRACK
16
8
8
16
8
8
DMA Controller
DMAREQ
DMAACK
I/O device controllers