# Computer Architecture and system programming

Kristoffer Klokker

2022

# Contents

# 1 Exercises

## 1.1 Benchmark

For the 40MHz processor which performed the instructions

| Instruciton type | Instruction count | Cycles per instruction |
|---|---|---|
| Integer artihemtic | 41,000 | 1 |
| Data transfer | 28,000 | 2 |
| Floating point | 25,000 | 2 |
| Control transfer | 6,000 | 2 |

### 1.1.1 Find the average CPI

$$\frac{1 \cdot 41000 + 2 \cdot 28000 + 2 \cdot 25000 + 2 \cdot 6000}{100000} = 1.59$$

CPI is the average cycles pr instruction. Therefore 4.5

### 1.1.2 Execution time

$$\begin{aligned} CPI &= 1.59 \\ I_c &= 100000 \\ \tau &= \frac{1}{f} = \frac{1}{40000000Hz} \\ T &= I_c \cdot CPI \cdot \tau \\ &= 1.59 \cdot 100000 \cdot \frac{1}{40000000Hz} \\ T &= 0.003975s \end{aligned}$$

### 1.1.3 MIPS

$$\begin{aligned} MIPS &= \frac{f}{CPI \cdot 10^6} \\ CPI &= 1.59 \\ f &= 40000000Hz \\ MIPS &= \frac{40000000Hz}{1.59 \cdot 10^6} \\ MIPS &= 25.16\frac{1}{s} \end{aligned}$$

## 1.2 Explain how a negative number is represnted in the following representation

- Sign-magnitude - The left most bit must be 1 which result in the rest being interpretated as negative
- Twos compliment - The left most bit is 1 to subtract the maximum value from the rest
- Biased - Bias is most usually half the range therefore a negative value is simply less half the maximum value

## 1.3 Represent the following in 8 bit twos compliment and sing magnitude

- 64 - 00100000
- -28 - twos 11100100 - sign 10011100

## 1.4 Convert from twos compliment to decimal

- 1100110 : -26
- 1011101 : -35

## 1.5 Show the calculations in 8 bit twos compliment

### 1.5.1 6+12

$$6 = 00000110$$
$$12 = 00001100$$
$$00000110 + 00001100 = 00010010$$

### 1.5.2 -6+12

$$-6 = 11111010$$
$$12 = 00001100$$
$$11111010 + 00001100 = 00000110$$

Overflow is ignored

### 1.5.3 6-12

$$\begin{aligned} 6 &= 00000110 \\ -12 &= 11110100 \\ 00000110 + 11110100 &= 11111010 \end{aligned}$$

### 1.5.4 -6-12

$$\begin{aligned} -6 &= 11111010 \\ -12 &= 11110100 \\ 11111010 + 11110100 &= 11101110 \end{aligned}$$

Overflow is ignored

## 1.6 Fill out the table for the most twos compliment addition

| Input | | | Output | | |
|---|---|---|---|---|---|
| $x_{n-1}$ | $y_{n-1}$ | $c_{n-2}$ | $z_{n-1}$ | $c_{n-1}$ | v |
| 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 0 | 0 | 1 |
| 0 | 1 | 0 | 1 | 0 | 0 |
| 0 | 1 | 1 | 0 | 1 | 0 |
| 1 | 0 | 0 | 1 | 0 | 0 |
| 1 | 0 | 1 | 0 | 1 | 0 |
| 1 | 1 | 0 | 1 | 1 | 0 |
| 1 | 1 | 1 | 1 | 1 | 1 |

Here $x_{n-1}$ and $y_{n-1}$ is the most signifact bits of the two addends.
$c$ is the carry bit and $z_{n-1}$ is the results most significatn bit.
$v$ is a bit singnaling overflow.
If can be seen in row 2 and the last row that:
Overflow occurs if and only if the carry into the addition of the MSBs isdifferent from the carry out of that addition

## 1.7 Convert 23 and 29 to 6 bit twos compliment and multiply using Booths algorithm

$$23 = 010111$$
$$29 = 011101$$
$$A = 0$$
$$Q_{-1} = 0$$
$$M = 010111$$
$$Q = 011101$$
$$count = 5$$

$$Q_0, Q_{-1} = 10$$
$$A = A - M = 101001$$
$$shift\ A = 101001\ Q = 011101\ Q_{-1} = 0$$
$$A = 110100$$
$$Q = 101110$$
$$Q_{-1} = 1$$
$$count = 4$$

$$Q_0, Q_{-1} = 01$$
$$A = A + M = 110100 + 010111 = 001011$$
$$shift$$
$$A = 000101$$
$$Q = 110111$$
$$Q_{-1} = 0$$
$$count = 3$$

$$Q_0, Q_{-1} = 10$$
$$A = A - M = 000101 - 011001$$
$$A = 000101 + 101001 = 101110$$
$$shift\ A = 101110\ Q = 110111\ Q_{-1} = 0$$
$$A = 110111$$
$$Q = 011011$$
$$Q_{-1} = 1$$
$$count = 3$$

$$
\begin{aligned}
Q_0, Q_{-1} &= 11 \\
shift\ A = 110111\ Q = 011011\ Q_{-1} &= 1 \\
A &= 111011 \\
Q &= 101101 \\
Q_{-1} &= 1 \\
count &= 2
\end{aligned}
$$

$$
\begin{aligned}
Q_0, Q_{-1} &= 11 \\
shift\ A = 111011\ Q = 101101\ Q_{-1} &= 1 \\
A &= 111101 \\
Q &= 110110 \\
Q_{-1} &= 1 \\
count &= 1
\end{aligned}
$$

$$
\begin{aligned}
Q_0, Q_{-1} &= 01 \\
A = A + M &= 111101 + 011001 \\
A &= 010100 \\
shift\ A = 010100\ Q = 110110\ Q_{-1} &= 1 \\
A &= 001010 \\
Q &= 011011 \\
Q_{-1} &= 0 \\
count &= 0
\end{aligned}
$$

$$010111 \times 011101 = AQ = 001010011011$$

# 2 Exercises

## 2.1 Convert from IEE 754 floating point to decimal

### 2.1.1 1 1000 0010 0010 0000 0000 0000 0000 000

$$
\begin{aligned}
&= (-1)^1 \cdot 2^{(10000010)_2 - 127} \cdot (1.00100000000000000000000)_2 \\
&= -1 \cdot 2^{130-127} \cdot 1.125 \\
&= -9.000
\end{aligned}
$$

#### 2.1.2 0 0111 1110 0000 1100 1100 1100 1100 110

$$\begin{aligned} &= (-1)^0 \cdot 2^{(01111110)_2 - 127} \cdot (1.00001100110011001100110)_2 \\ &= 1 \cdot 2^{126-127} \cdot 1.04999995231628417969 \\ &\approx 0.5249999760 \end{aligned}$$

#### 2.1.3 0 1000 0000 1100 1100 1100 1100 1100 110

$$\begin{aligned} &= (-1)^0 \cdot 2^{(10000000)_2 - 127} \cdot (1.10001100110011001100110)_2 \\ &= 1 \cdot 2^{128-127} \cdot 1.79999995231628417969 \\ &\approx 3.599999904 \end{aligned}$$

### 2.2 Convert from decimal to IEEE 754 floating point

#### 2.2.1 -720

$$\begin{aligned} 720 &= 1011010000_2 \\ 1.011010000_2 &\cdot 2^{9_{10}+127} \\ 1.011010000_2 &\cdot 2^{136} \\ 10001000_2 &= 117_{10} \\ 110001000011010000 & \end{aligned}$$

#### 2.2.2 0.645

$$\begin{aligned} 0.645 &= 0.10100101000111101011 10_2 \\ 1.01001010001111010111 0_2 &\cdot 2^{-1_{10}+127} \\ 1.01001010001111010111 0_2 &\cdot 2^{126} \\ 01111110_2 &= 126_{10} \\ 00111111001001010001111010111 0 & \end{aligned}$$

### 2.3 Which numbers can be exactly represented in IEE 754

- 17.0 - representable inside the range
- -1 - representable inside the range

- $\frac{7}{16}$ - representable inside range since equal 0.4375
- $\frac{1}{3}$ - not representable due to infinite
- $\pi$ - not representable due to infinite
- $5.4321 \cdot 10^6$ - representable inside range
- $6.022 \cdot 10^{23}$ - representable inside range

## 2.4 Let $C$ and $D$ denote two number in IEEE 754 single-precision floating point format

$$C = 01000010101010100000000000000000$$
$$D = 11000010011111100000000000000000$$

### 2.4.1 What are the decimal values of $C$ and $D$

$$C = (-1)^0 \cdot 2^{(10000101)_2 - 127} \cdot (1.01010100000000000000000)_2$$
$$C = 1 \cdot 2^{133-127} \cdot 1.328125$$
$$C = 85.000000$$

$$D = (-1)^1 \cdot 2^{(10000100)_2 - 127} \cdot (1.11111100000000000000000)_2$$
$$D = -1 \cdot 2^{132-127} \cdot 1.984375$$
$$D = -63.500000$$

### 2.4.2 Make the addition of floating points

$$X = 01000010101010100000000000000000$$
$$Y = 11000010011111100000000000000000$$
$$X_e = 6$$
$$X_b = 1.01010100$$
$$Y_e = 5$$
$$Y_b = 0.11111100$$
$$Y = 1.11111100 \cdot 2^5$$
$$Y = 0.11111110 \cdot 2^6$$
$$S = x_b - Y_b = 0.01010110 \cdot 2^6$$
$$S = 1.010110 \cdot 2^4$$
$$S_e = 4 + 127 = 131 = 10000011_2$$
$$S = 01000001101011000000000000000000$$

## 2.5 Create a truth table for the following algebra expression

| $A$ | $B$ | $C$ | $(A + \neg B + C)$ | $(\neg A + B + \neg C)$ | $(A + \neg B + C)(\neg A + B + \neg C)$ |
|---|---|---|---|---|---|
| 1 | 0 | 0 | 1 | 1 | 1 |
| 1 | 1 | 0 | 1 | 1 | 1 |
| 0 | 0 | 0 | 1 | 1 | 1 |
| 0 | 1 | 0 | 0 | 1 | 0 |
| 1 | 0 | 1 | 1 | 0 | 0 |
| 1 | 1 | 1 | 1 | 1 | 1 |
| 0 | 0 | 1 | 1 | 1 | 1 |
| 0 | 1 | 1 | 1 | 1 | 1 |

# 3 Exercises

## 3.1 Consider these two programs

```
01 | for (i=1; i<n; i++) {
02 |    z[i]=x[i]-y[i]
03 |    z[i]=z[i]*z[i]
04 | }
```

```
01 | for (i=1; i<n; i++) {
02 |    z[i]=x[i]-y[i]
03 | }
04 | for (i=1; i<n; i++) {
05 |    z[i]=z[i]*z[i]
06 | }
```

The function of the programs are finding the elemental difference between list x and y and squaring it.
This is done most efficient by the first program since the list can stay in cache unlike the second program which loads the list x, y and z and then loads z again.

## 3.2 Consider the cache with an access time of 5 ns and a hit ratio of $H = 0.9$. The memory access time alone is 100ns.

### 3.2.1 What is the average access time for this system?

$0.9 \cdot 5ns + 0.1 \cdot (100ns + 5ns) = 15ns$

### 3.2.2 Suppose the cache access time is increased to 6 ns. What is the minimum hit raio needed in order to not increase the average access time?

$x \cdot 6ns + (1 - x) \cdot (100ns + 6ns) = 15ns$
$x = 0.91$

### 3.2.3 Suppose the cache access time is instead increased to 10 ns. What is the minimum hit ratio needed in order to not increase the average access time?

$x \cdot 10ns + (1 - x) \cdot (100ns + 10ns) = 15ns$
$x = 0.95$

## 3.3 What is the average time in the following system

9ns cache
80ns main memory
8ms from disk to main memory
cache miss rate 9%
main memory mis rate 30%
$0.91 \cdot 9ns + 0.09 \cdot ((0.7 \cdot 80ns + 0.3 \cdot (80ns + 8ms)) + 9ns) = 0.216ms$

## 3.4 A cache has a line size of 64 bytes. To determine which byte within a cache line an address points to, how many bits are in the Offset field?

$\log_2(64) = 6$

## 3.5 A two-way set-associative cache in a word addressable machine consist of 128 cache lines divided into several sets. The main memory contain 8 K (8192) block of size 256 words. Show and explain the format of main memory addresses

cache size: 128 lines
MM size: $8192 \cdot 256 = 2097152$ words
block size: 256 words
block size $= 2^x = 256 \rightarrow x =$ offset $= 8$
MM size $= 2^z = 2097152 \rightarrow z =$ physical address bits $= 21$
Since the set contains 2 lines since it is a two-way set
Number of lines $= 128 = 2^y \rightarrow y = 7$
Therefore set number is line number /2 $2^7/2 = 2^6$ making the set number equal to 6.
Tag size = physical address bits - offset - set number = 21 - 8 - 6 = 7

## 3.6 Calculate the cache

En 32-bit maskine har en cache med 32 indgange (”cache lines”), hver på 16 bytesdata. En adresse er på 32 bit og adresserer de enkelte bytes.Cachen er organiseretsom en 2-vejs cache (2-way set associative).
Line numbers: 32 lines $= 2^5$
MM address: 32 bit system $= 2^32$
Block size: 16 byte $= 2^4$
offset = log(blocksize = 4 bit
set number bits = index = log(line numbers)/2=4 bit
tag = log(MM) - offset - index = 24

### 3.7 Using the same cache system from last exercise check if the following gives a hit or miss

| indeks | V | T | D | V | T | D |
|---|---|---|---|---|---|---|
| 0 | 1 | 2 | x | 1 | AA | x |
| 1 | 1 | 29FF | x | 1 | 7600 | x |
| 2 | 0 | F5 | x | 1 | 4 | x |
| 3 | 1 | 39E0 | x | 1 | 1210 | x |
| 4 | 1 | 2221 | x | 0 | 443B | x |
| 5 | 1 | 60 | x | 0 | 1BBB | x |
| 6 | 0 | 60 | x | 0 | 61 | x |
| 7 | 1 | 1210 | x | 0 | 61 | x |
| 8 | 0 | 60 | x | 0 | 61 | x |
| 9 | 0 | 76 | x | 1 | 61 | x |
| 10 | 0 | 76F | x | 1 | D59 | x |
| 11 | 0 | 0 | x | 0 | 0 | x |
| 12 | 1 | 1210 | x | 1 | 7701 | x |
| 13 | 1 | 1210 | x | 0 | 222A | x |
| 14 | 1 | 1210 | x | 0 | 223C | x |
| 15 | 1 | 1210 | x | 1 | 100A | x |

| Tilfælde | Læs/skriv | Antal byte | Adresse |
|---|---|---|---|
| A: | læs | 4 | 00076FA4 |
| B: | skriv | 4 | 00121070 |
| C: | læs | 2 | 000D59C6 |
| D: | læs | 4 | 00000428 |
| E: | læs | 2 | 00021080 |

A index 10 no longer valid hit
B index 7 still valid hit
C index 12 miss
D index 2 still valid
E index 8 miss

# 4 Exercises

## 4.1 In virtually all systems that include DMA modules, DMA to main memory is given higher priority that CPU access to main memory. Why?

Since it works on stealing cycles, in case of CPU having higher priority, it would not be able to steal cycles.

## 4.2 If we were to examine the timing diagram for the 8237A, we would find that once a block transfer begins, it takes three bus clock cycles per DMA cycle. During the DMA cycle, the 8237A transfers one byte of information between memory and I/O device

### 4.2.1 Suppose we clock the 8237A at a rate of 8 MHz. How long does it take to transfer one byte?

1 byte transfer between ememory and IO device,3 clock cycles for one DMA cycle.

$$\frac{3}{\frac{1000000}{s}} = 0.000003s$$

### 4.2.2 What would be the maximum attainable data transfer rate?

It took 0.000003s for a byte transfer making the transfer rate

$$\frac{1byte}{0.000003s} = 333kByte$$

### 4.2.3 Assume that the memory is not fast enough and we have to insert four wait states per DMA cycle. What will be the actual transfer rate?

$$\frac{4}{\frac{1000000}{s}} = 0.000004s$$

$$\frac{1byte}{0.000004s} = 250kByte$$

## 4.3 If the last operation performed on a computer with an 8-bit word was an addition in which the two operands were 0010 0100 and 0001 0001,what would be the value of the following flags?

The result will be 00110101

- Carry - 0
- Zero - 0
- Overflow - 0
- Sign - 0 - even if treated as 2s compliement both are positive so result is also positive
- Even parity - 1 - the result has an even number of 1's
- Half-carry - 0 - The left most bit dit not change

## 4.4 Repeat the last exercise with -2 and +3

1111 1101 -2
0000 0011 +3
0000 0000 0

- Carry - 1
- Zero - 1
- Overflow - 1
- Sign - 0
- Even parity - 1
- Half-carry -1

## 4.5 Assuming that each pipeline stage takes one cycle, calculate the number of cycles needed to complete executing 10 instructions in a four-stage pipeline. Calculate the speed-up factor for this pipeline compared to serial implementation without a pipeline

The speedup factor

$$\frac{10 \cdot 4}{4 + (10 - 1)} = 3.08$$

For the none pipeline it would take 40 cycles, for the pipeline it would only take 13

## 4.6 A CPU executes programs in the following 6-stage pipeline

1. FI- fetch
2. DI - decode
3. CO - calculate operand
4. FO - fetch operand
5. EI - execute
6. WO - write out

Each stage takes 1 clock cycle. However, the memory bandwidth is limited, such that only a single stage can access memory at a time. That is, only one of FI, FO, and WO can access memory in each cycle. In the following, ignore the cache, and assume every operand is storedin memory.

### 4.6.1 For the following program, draw a timing diagram (like Fig. 16.10, with time and instructions as axes) to represent the instruction pipeline operation. Mark bubbles in the pipeline with DH if the stall is becauseof a data hazard (the data is not yet available), or with SH if it's because of a structural hazard (memory can't be accessed).

ADD A,B
MOV D,B
SUB, C,A

ADD A,C

| | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| ADD | FI | DI | CO | FO | EI | WO | | | | | | |
| MOV | | FI | DI | CO | FO | EI | WO | | | | | |
| SUB | | | FI | DI | CO | DH | FO | EI | WO | | | |
| ADD | | | | FI | DI | CO | DH | DH | DH | FO | EI | WO |

### 4.6.2 Reorder the instruction to speed up

They can be reordered in a better way since the SUB need the result from the add which is just done when the sub is started, and for the last add it has to wait for the sub anyway

## 4.7 Analyze assembly code

`long int *doStuff(long int *first, long int *last)`
Below you find the assembly code for the function.

```
1  doStuff:
2     pushq   %rbp
3     movq    %rsp, %rbp
4     subq    $16, %rsp
5     movq    %rdi, -8(%rbp)
6     movq    %rsi, -16(%rbp)
7     jmp     label2
8  label5:
9     movq    -8(%rbp), %rax
10    movq    (%rax), %rax
11    movq    %rax, %rdi
12    call    pred
13    cmpq    $0, %rax
14    je      label3
15    movq    -8(%rbp), %rax
16    jmp     label4
17 label3:
18    addq    $8, -8(%rbp)
19 label2:
20    movq    -8(%rbp), %rax
21    cmpq    -16(%rbp), %rax
22    jne     label5
23    movq    -16(%rbp), %rax
24 label4:
25    movq    %rbp, %rsp
26    popq    %rbp
27    ret
```

### 4.7.1 identify the liens with the prolog and epilog of the function

rbp is pushed on stack at line 2 and popped at 26

### 4.7.2 Where is first and last stored at first

the order goes
%rdi, %rsi, %rdx, %rcx, %r8 and %r9
So the two first since it a 64 bit long parameter is in the r form.

### 4.7.3 The function uses the stack for storing data. Show a table of the stack-frame for the function. That is, a list of the entries on the stack created by the function, what they each are used for, and their address relative to the base pointer

MAIN EPB
Return address of MAIN
RDI
RSI

### 4.7.4 The function calls another function,pred, for which we only know that it returns either 0 or 1. How many arguments are passed to pred, assuming the function follows the normal calling convention on 64-bit Linux systems?

RDI

### 4.7.5 It looks like there is a loop in the function. Identify the lines where the condition for this loop is calculated and used, and describe the loop condition, e.g., in terms of the C variables

The for loop can be identified as label 2 being the condition, label 3 being the addition of the for loop and label 5 being the content of the for loop

### 4.7.6 Describe the functionality of the function, e.g., by expressing it in higher-level pseudo code or as C code, and a short textual description

First the rdi and rsi is added to the stack.
It then check if the first paramter is equal to the last. if not it goes to the for loop
The content of the loop takes the address of first, an parses it to pred and if it return to zero it adds one to first and loops again.
Otherwise it ends the functions

# 5 Exercise

## 5.1 Consider the follwoing assembly language program

1. Load R2, (R4)
2. Ble R0 R2 L2
3. Add R2, R2, 6
4. Move R1, R2
5. Load R1, (R5)

This program includes WAW, RAW and WAR dependencies how these.
RAW - 4 - 5
WAW - 3- 4
WAR - 1 - 3

## 5.2 Let $a$ be the percentage of program code that can be executed simultaneously by $n$ processors in a computer system. Assume that the remaining code must be executed sequentially by a single processor. Each processor has an execution rate of $x$ MIPS

### 5.2.1 Derive an expression for the effective MIPS rate when using the system for exclusive execution of this program, in terms of $n$, $a$, and $x$

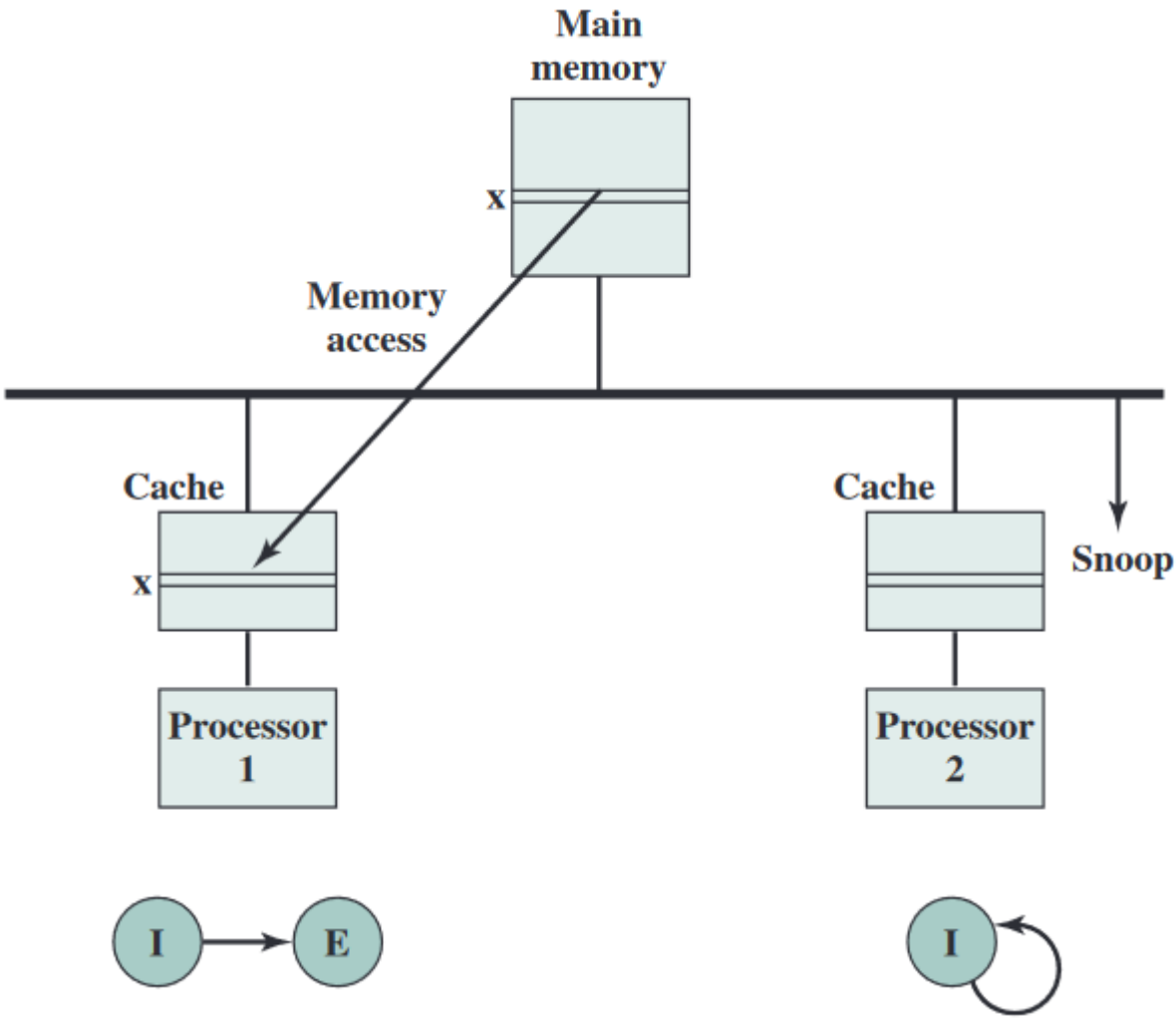
$$a \cdot n \cdot x + (1 - a) \cdot x$$

### 5.2.2 If $n = 16$ and $x = 4$ MIPS, determine the value of $a$ that will yield a system performance of 40 MIPS

$$a \cdot 16 \cdot 4MIPS + (1 - a) \cdot 4MIPS = 40MIPS$$

$$a = \frac{3}{5}$$

5.3 Consider a situation in which two processors in an SMP configuration, over time, require access to the same line of data from main memory. Both processors have a cache and use the MESI protocol. Initially, both caches have an invalid copy of the line. Figure 20.13 depicts the consequence of a read of line x by Processor P1. If this is the start of a sequence of accesses, draw the subsequent figures for the following sequence



#### 5.3.1 P2 reads x

P2 also just reads x into memory and set as shared in both p1 and p2

#### 5.3.2 P1 writes to x (label the line in P1s cache x')

Writes to x in cache, invalidates the p2 cache x and changes its own to modified

#### 5.3.3 P1 writes to x (label the line in P1s cache x")

Write again withotu changing anything

#### 5.3.4 P2 reads x

Signal p1 to write its modified version to memory and then change it to shared and reads it from memory as shared

# 6 Exercises

## 6.1 You find yourself debugging a program which sends and receives commands over the Internet. Due to the unreliability of the network connection the program uses error-correction codes for messages. Specifically, each 8-bit message is encoded as a 12-bit Hamming code

### 6.1.1 The program would like to send the following 8 bits of data, written from left to right as d1,d2. . . d8: 1111 0001. Compute the corresponding 12-bit Hamming code, and indicatethe check bits

1000

**6.1.2 The program is receives the following 12 bits in response, written from left to right as b1,b2. . . b12: 1001 1100 0001. Assuming that at most 1 single-bit error has occurred,compute the 8-bit message**

## 6.2 Boolean Algebra

### 6.2.1 Give the Karnaugh Maps for X, Y and Z as a table

| X | | C | |
|---|---|---|---|
| | | 0 | 1 |
| AB | 00 | | |
| | 01 | | |
| | 10 | 1 | 1 |
| | 11 | 1 | 1 |

Four circling

$$X = A$$

Four circling

| Y | | C | |
|---|---|---|---|
| | | 0 | 1 |
| AB | 00 | | |
| | 01 | 1 | 1 |
| | 10 | 1 | 1 |
| | 11 | | |

$$Y = \overline{A}B + A\overline{B}$$

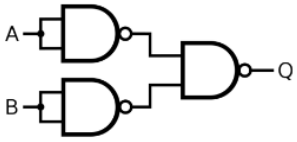
Only one circles

| Z | | C | |
|---|---|---|---|
| | | 0 | 1 |
| AB | 00 | | 1 |
| | 01 | 1 | |
| | 10 | | 1 |
| | 11 | 1 | |

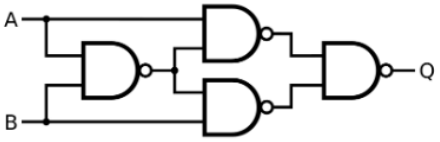
$$Z = \overline{AB}C + \overline{AC}B + \overline{B}AC + \overline{C}AB$$

## 6.3 Give truth table for circuits

C1




C2




C1 - OR gate

| A/B | 0 | 1 |
|---|---|---|
| 0 | 0 | 1 |
| 1 | 1 | 1 |

C2 - XOR

| A/B | 0 | 1 |
|---|---|---|
| 0 | 0 | 1 |
| 1 | 1 | 0 |