Homework 1

1.7 [15] <§1.6> Compilers can have a profound impact on the performance of an application. Assume that for a program, compiler A results in a dynamic instruction count of 1.0E9 and has an execution time of 1.1 s, while compiler B results in a dynamic instruction count of 1.2E9 and an execution time of 1.5 s.

a. Find the average CPI for each program given that the processor has a clock cycle time of 1 ns. 3 points

b. Assume the compiled programs run on two different processors. If the execution times on the two processors are the same, how much faster is the clock of the processor running compiler A's code versus the clock of the processor running compiler B's code? 3 points

c. A new compiler is developed that uses only 6.0E8 instructions and has an average CPI of 1.1. What is the speedup of using this new compiler versus using compiler A or B on the original processor? 3 points

a.

CPI_A = 周期数/指令数 = (1.1s/1.0ns) / 1.0E9 = 1.1

CPI_B = 周期数/指令数 = (1.5s/1.0ns) / 1.2E9 = 1.25

b.

CPU timeA = 周期数 A / 时钟频率 A

CPU timeB = 周期数 B / 时钟频率 B

We have CPU timeA = CPU timeB

Frequency A / Frequency B = (1.1s/1.0ns) /(1.5s/1.0ns) = 1.1/1.5 = 0.73

So the clock frequency of the processor running A is 0.73 times that of B

C.

Cycle = CPI * 指令数= 6.6*10^8

A: (1.1s/1.0ns) = 1.1 *10 ^9

B: (1.5s/1.0ns) = 1.5 *10 ^9

11/ 6.6 = 1.666

15/ 6.6 = 2.2727

1.9 Assume for arithmetic, load/store, and branch instructions, a processor has CPIs of 1, 12, and 5, respectively. Also assume that on a single processor a program requires the execution of 2.56E9 arithmetic instructions, 1.28E9 load/store instructions, and 256 million branch instructions. Assume that each processor has a 2 GHz clock frequency.

Assume that, as the program is parallelized to run over multiple cores, the number of arithmetic and load/store instructions per processor is divided by $0.7 \times p$ (where p is the number of processors) but the number of branch instructions per processor remains the same.

1.9.1 [5] < §1.7> Find the total execution time for this program on 1, 2, 4, and 8 processors, and show the relative speedup of the 2, 4, and 8 processors result relative to the single processor result. 3 points

1 processor: 2.56E9 arithmetic, 1.28E9 load/store, 2.56E8 br.

Cycles: 2.56E9 *1 + 1.28E9*12 + 2.56E8 *5 = 1.884E 10

Execution time: 1.884E 10 / 2E9 = 9.42s

2 processors: 2.56E9 arithmetic, 1.28E9 load/store, 2.56E8 br.

Cycles: 2.56E9 *1/1.4 + 1.28E9*12/1.4 + 2.56E8 *5 = 1.408E 10

Execution time: 1.408E 10 / 2E9 = 7.04s

7.04s/9.42s = 74.7%

2 cores are 1.34 times faster than one core

4 processors: 2.56E9 arithmetic, 1.28E9 load/store, 2.56E8 br.

Cycles: 2.56E9 *1/2.8 + 1.28E9*12/2.8 + 2.56E8 *5 = 7.68E 9

Execution time: 7.68E 9 / 2E9 = 3.84s

3.84s /9.42s = 40.76%

4 cores are 2.45 times faster than one core

8 processors: 2.56E9 arithmetic, 1.28E9 load/store, 2.56E8 br.

Cycles: 2.56E9 *1/5.6 + 1.28E9*12/5.6+ 2.56E8 *5 = 4.48E 9

Execution time: 4.48E 9 / 2E9 = 2.24s

2.24s /9.42s = 23.7%

Eight cores are 4.2 times faster than one core.

1.9.2 [10] <§§1.6, 1.8> If the CPI of the arithmetic instructions was doubled, what would the impact be on the execution time of the program on 1, 2, 4, or 8 processors? 3 points

1.9.3 [10] <§§1.6, 1.8> To what should the CPI of load/store instructions be reduced in order for a single processor to match the performance of four processors using the original CPI values? 3 points

1 processor: 2.56E9 arithmetic, 1.28E9 load/store, 2.56E8 br.

Cycles: 2.56E9 *2 + 1.28E9*12 + 2.56E8 *5 = 2.176E 10

2.176E 10 / 1.884E 10 = 1.155

So the execution time of the program on 1 processor increase 15.5%

2 processors: 2.56E9 arithmetic, 1.28E9 load/store, 2.56E8 br.

Cycles: 2.56E9 *2/1.4 + 1.28E9*12/1.4 + 2.56E8 *5 = 1.59E 10

1.59E 10 / 1.408E 10 = 1.13

So the execution time of the program on 2 processor increase 13%

4 processors: 2.56E9 arithmetic, 1.28E9 load/store, 2.56E8 br.

Cycles: 2.56E9 *2/2.8 + 1.28E9*12/2.8 + 2.56E8 *5 = 8.594E 9

8.594E 9 / 7.68E 9= 1.12

So the execution time of the program on 4 processor increase 12%

8 processors: 2.56E9 arithmetic, 1.28E9 load/store, 2.56E8 br.

Cycles: 2.56E9 *2/5.6 + 1.28E9*12/5.6+ 2.56E8 *5 = 4.937 E 9

4.937 E 9 / 4.48E 9 = 1.102

So the execution time of the program on 8 processor increase 10.2%

In conclusion, the more processors we have, the less impact the CPI increment has on the execution time.

1.9.3

4 processors: 2.56E9 arithmetic, 1.28E9 load/store, 2.56E8 br.

Cycles: 2.56E9 *1/2.8 + 1.28E9*12/2.8 + 2.56E8 *5 = 7.68E 9

4 cores are as fast as one core. Since they have the same clock frequency, they should have the same clock cycles.

1 processor: 2.56E9 arithmetic, 1.28E9 load/store, 2.56E8 br.

Cycles: 2.56E9 *1 + 1.28E9*x + 2.56E8 *5 = 7.68E 9

1.28E9*x = 3.84

X = 3 =CPI = 3

3/12 = 25%

Thus, we need to change the cpi of L/S to 25% of the original.

1.14 Assume a program requires the execution of 50×10^6 FP instructions, 110×10^6 INT instructions, 80×10^6 L/S instructions, and 16×10^6 branch instructions. The CPI for each type of instruction is 1, 1, 4, and 2, respectively. Assume that the processor has a 2 GHz clock rate.

1.14.1 [10] <§1.10> By how much must we improve the CPI of FP instructions if we want the program to run two times faster? 3 points

1.14.2 [10] <§1.10> By how much must we improve the CPI of L/S instructions if we want the program to run two times faster? 3 points

1.14.3 [5] <§1.10> By how much is the execution time of the program improved if the CPI of INT and FP instructions is reduced by 40% and the CPI of L/S and Branch is reduced by 30%? 3 points

Float point 50 *10 ^6

Int 110 *10 ^6

L/S 80 *10 ^6

Branch 16 *10 ^6

1.14.1

Cycle: 50 *10 ^6 + 110 *10 ^6 + 4* 80 *10 ^6 + 2* 16 *10 ^6 = 5.12 *10^8

Now Cycle = 2.56 *10^8, we need to reduce 2.56 *10^8 cycles.

However, FP cycle = 50 *10 ^6 < 2.56 *10^8 cycles. So we cannot let the program run two times faster by improving the CPI of FP instruction only.

1.14.2

Cycle: 50 *10 ^6 + 110 *10 ^6 + 4* 80 *10 ^6 + 2* 16 *10 ^6 = 5.12 *10^8

Now Cycle = 2.56 *10^8

L/S: 320 – 256 = 64

The CPI of L/S = 0.8, 0.8 /4 = 0.2

Thus, we need to change the cpi of L/S to 20% of the original.

1.14.3

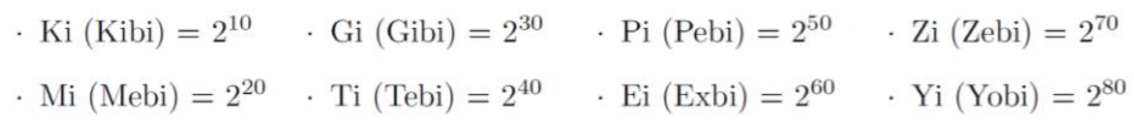
Now Cycle: 0.6* 50 *10 ^6 + 0.6* 110 *10 ^6 + 0.7*4* 80 *10 ^6 + 0.7*2* 16 *10 ^6 = 3.424 *10^8

Thus, we have changed the execution time to 66.875% of the original

1. How many bits do we need to represent a variable that can only take on the values 0, π or e? (3 points)

2 bits

2. If we need to address 3 TiB of memory and we want to address every byte of memory, how long does an address need to be? (3 points)



· Ki (Kibi) = 2^{10}	· Gi (Gibi) = 2^{30}	· Pi (Pebi) = 2^{50}	· Zi (Zebi) = 2^{70}
· Mi (Mebi) = 2^{20}	· Ti (Tebi) = 2^{40}	· Ei (Exbi) = 2^{60}	· Yi (Yobi) = 2^{80}

3* 2^40 bytes, 40 +2 = 42 bits

3. If the only value a variable can take on is e, how many bits are needed to represent it. (3 points)

1 bit

4. Assume an 8-bit integer and answer each one for the case of an unsigned number, biased number with a bias of -127, and two's complement number, indicating if it cannot be answered with a specific representation.

(1) What is the largest integer? The largest integer's representation + 1?

(a) [Unsigned] (3 points)

255 = 0b1111 1111, 0 = 0b0000 0000

(b) [Biased] (3 points)

128= 0b1111 1111, -127 = 0b0000 0000

(c) [Two's Complement] (3 points)

0111 1111 = 127, -128 = 0b1000 0000

(2) How do you represent the numbers 0, 1, and -1?

(a) [Unsigned] (3 points)

0 = 0b0000 0000, 1 = 0b0000 0001. -1 cannot be answered with a specific representation.

(b) [Biased] (3 points)

1 = 1000 0000, 0= 0111 1111, -1 = 0111 1110

(c) [Two's Complement] (3 points)

0b0000 0000 = +0, - 0 = 1000 0000,1= 0b0000 0001, -1 = 0b1111 1111

(3) How do you represent 17, -17?

(a) [Unsigned] (3 points)

17 = 0b0001 0001, -17 = cannot be answered with a specific representation.

(b) [Biased] (3 points)

17 = 1001 0000, -17 = 0110 1110

(c) [Two's Complement] (3 points)

17 = 0b0001 0001, -17 = 0b1110 1111

(4) What is the largest integer that can be represented by any encoding scheme that only uses 8 bits? (3 points)

Because bias can be added, there is no maximum.

(5) Prove that the two's complement inversion trick is valid (i.e. that x and x + 1 sum to 0). (3 points)

We have gain a num y which every bit is 1 by adding x with (x's inversion)

Then y plus one equals 0.

5. If we only have shift registers and adders, how to implement a system
    1. that can multiply any integer by 3? (7 points)
    2. if we change the multiplicand to 7? (Can you design more than one structure? Which is better?) (7 points)
    3. if the integer will be divided by 3? (7 points)

    Hint: 2's complement can be used to solve the above questions.

1. x<< 1 +x

2. x <<3 + (–x) use the 2's complement to gain the -x

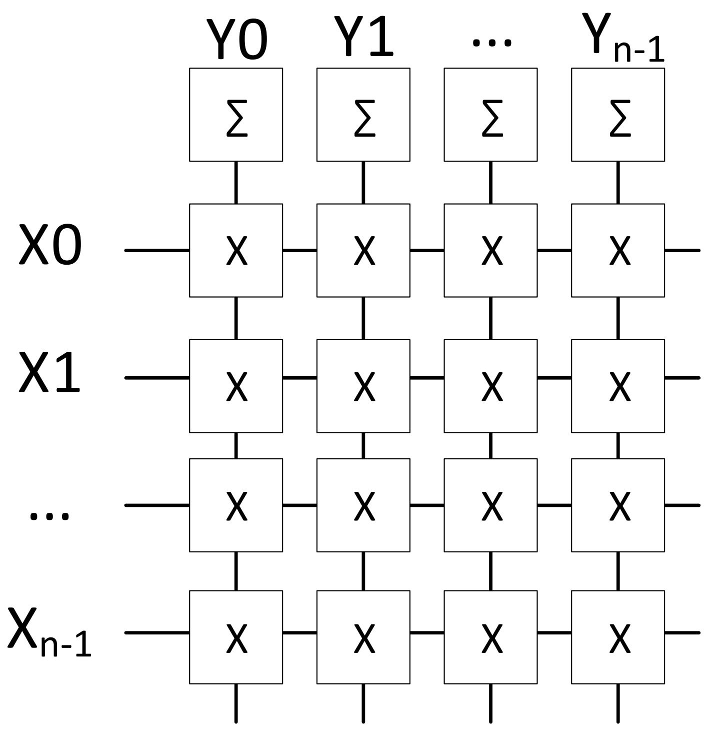
3. $\sum_{i=1}^{15}(x \ll (2i))$ $\quad because \quad \lim \sum_{i=1}^{\backslash infity} \frac{1}{4^i} = \frac{1}{3}$

X /3 = x/4 + x/16 + x/64 + …. = x<<2 + x<< 4 + x<<6 + ….

Because integer have 32bits

X/3 = x<<2 + x<< 4 + … + x<<30

6. A system below is designed for the 16-b fixed-point MAC operation. It has an n*n array of the multipliers. The operands A ($a_{15}\ldots a_4a_3a_2a_1a_0$) and B ($b_{15\ldots}b_5b_4b_3b_2b_1b_0$) are the two inputs of the multiplier. Output Y has 32 bits. Assuming operand B is a constant and stored in the multiplier. The multiplication is based on the shift and add operation (… a_2*B*2^2+ a_1*B*2^1+ a_0*B*2^0, where *2^n is achieved by the shift operation). We have the following two assumptions:
    1) data B follow the Gaussian distribution and their mean value is 0;
    2) when a bit in B is 0, there is no power consumption.
    这个系统 设计一个 16bit mac 操作，有 n*n 乘法器，
    To minimize the power consumption, what kind of binary representation should be used? Please draw a block diagram of the multiplier and explain why it helps to reduce the power consumption. (10 points)
    应该用哪种二进制表示？ 我们应该让 1 的概率小.
    *Hint: Minimizing the power consumption is to minimize the ratio of "1" s in data B.



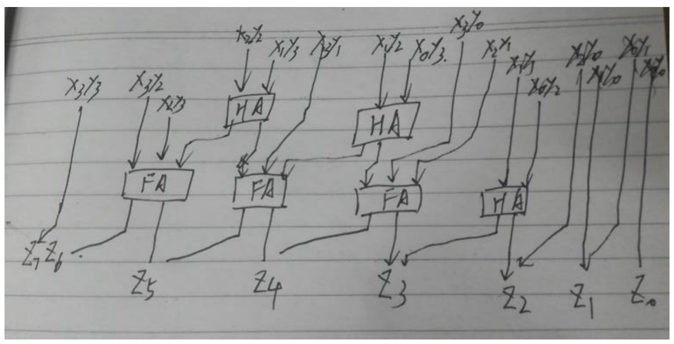
An unsigned number cannot represent a negative number, so we select representation from bias or two's complement.

When bi equals 0, xj *bi would not consume power, data B is likely near 0 because data B follow the Gaussian distribution and their mean value is 0;

When B > 0, each bit of B is more likely to be zero instead of one. B's inversion equals B. If we use bias, we may need to add some 1.

When B < 0, each bit of B is more likely to be 1 instead of 0, but B's complement is more likely to be zero instead of one. Thus, two's complement is better than bias. That is why we choose two's complement representation.

下图这样可以节省需要的硬件，减少了传播延时,在乘法位数较多时比进位保留结构快得多.


HA
FA
Z_7 Z_6
Z_5
Z_4
Z_3
Z_2
Z_1
Z_0