CS18BTECH11039_CA_HW2

RAJ PATIL

This document is generated by $\LaTeX$

Abstract

Contains the answers and comments to questions provided in homework 2 of CS2323.

Contents

1 Question 1

Given:-

$$\begin{aligned} Total\ bits &= T \\ (E)\ Exponent\ bits &= E \\ (S)\ Significand\ bits &= T - E - 1 \\ (B)\ bias\ &= 2^{E-1} - 1 \end{aligned}$$

1.1 A) :- Normal Numbers

for the largest Normal number, Exponent will be set to set to all ones except the LSB. Significand will be set to all digits as 1

$$\begin{aligned} Exponent &= 2^E - 2 \\ Significand &= 1 + 1 - 2^{-S} \\ Corresponding\ number &= (2 - 2^{-S}) * 2^{2^E - 2 - B} \\ &= (2 - 2^{-S}) * 2^{2^E - 2 - 2^{E-1} + 1} \\ &= (2 - 2^{-S}) * 2^{2^{E-1} - 1} \end{aligned}$$

for the smallest Normal number, Exponent will be 1 and significand is set to 0

$$\begin{aligned} Exponent &= 1 \\ Significand &= 0 \\ Corresponding\ number &= (1 + 0) * 2^{1-B} \\ &= 2^{1 - 2^{E-1} + 1} \\ &= 2^{2 - 2^{E-1}} \end{aligned}$$

1.2 A) :- Denormal Numbers

for the Largest denormal number, Exponent is set to 0, Significand will be all 1s

$$\begin{aligned} Exponent &= 0 \\ Significand &= 0 + 1 - 2^{-S} \\ Corresponding\ number &= (1 - 2^{-S}) * 2^{1-B} \\ &= (1 - 2^{-S}) * 2^{1 - 2^{E-1} + 1} \\ &= (1 - 2^{-S}) * 2^{2 - 2^{E-1}} \end{aligned}$$

for the smallest denormal number, Exponent is 0, significand will be all set to 0 except the LSB

$$\begin{aligned} Exponent &= 0 \\ Significand &= 0 + 2^{-S} \\ Corresponding\ number &= (2^{-S}) * 2^{1-B} \\ &= (2^{-S}) * 2^{1 - 2^{E-1} + 1} \\ &= (2^{-S}) * 2^{2 - 2^{E-1}} \end{aligned}$$

1.3 B) :- FP16

Given:-

$$\begin{aligned} T &= 16 \\ E &= 5 \\ sign &= 1 \\ S &= 16 - 5 - 1 \\ &= 10 \\ B &= 15 \end{aligned}$$

FP 16		
	Normal	Denormal
Largest	$(2 - 2^{-10}) * 2^{15}$	$2^{-14} - 2^{-24}$
Smallest	2^{-14}	2^{-24}

1.4 B) :- bfloat16

Given:-

$$\begin{aligned} T &= 16 \\ E &= 8 \\ sign &= 1 \\ S &= 16 - 8 - 1 \\ &= 7 \\ B &= 127 \end{aligned}$$

bfloat16		
	Normal	Denormal
Largest	$(2 - 2^{-7}) * 2^{127}$	$2^{-126} - 2^{-133}$
Smallest	2^{-126}	2^{-133}

1.5 C)

Generic case:-

given usual notation, the difference between the smallest and the second smallest normal number is given as :-

$$(1 + 2^{-S}) * 2^{1-B} - 2^{1-B} = 2^{-S} * 2^{1-B} = 2^{1-B-S}$$

Corresponding values as follows:-

$$\begin{aligned} FP16 &= 2^{1-15-10} \\ &= \mathbf{2^{-24}} \\ Bfloat16 &= 2^{1-127-7} \\ &= \mathbf{2^{-133}} \end{aligned}$$

1.6 D)

Range of bfloat16 is almost similar to that of FP32 but that of FP16 is very limited

In bfloat16 , the precision varies a lot $(2^{-126} to\ 2^{120})$ **as we observe larger normal numbers, whereas this variance is low for FP16**$(2^{-14}\ to\ 2^{-5})$

Relative precision of bfloat16 is lower than that of FP16 (no of precision bits in bfloat16 are 7*log(2) (approx 3) whereas this number is 10*log(2) (approx 4) in the case of FP16)

1.7 E)

range of bfloat16 is almost similar to that of FP32 because the number of exponents bits are the same for them(8),hence its easy to convert between the two just truncate or add significand bits correspondingly

2 Question 2

$$\begin{aligned}
&Given\\
virtual\ address\ size &= 48\ bits\\
physical\ memory\ size &= 2GB = 2*2^{30} = 2^{31}B\\
physical\ address\ size &= 31\ bits\\
page\ size &= 2*2^{10}B = 2^{11}B\\
page\ offset &= 11\ bits\\
no.\ of\ frame\ bits &= 31-11 = 20\\
no.\ of\ page\ number\ bits &= 48-11 = 37\\
no.\ of\ bits\ in\ one\ entry\ of\ TLB &= 37+20 = 57\ (ignoring\ other\ bits)\\
total\ entries\ in\ TLB &= 64\\
size\ of\ TLB &= 57*64\ bits = 57*8\ Bytes\\
&= \textbf{456 Bytes}
\end{aligned}$$

3 Question 3

the reach of a TLB is the net amount of memory it can reach Reach of TLB here:-

$$4KB*128+2MB*32+2GB*8 = (2^{19}+2^{26}+2^{34})B = 1.724*10^{10}B$$

taking log for writing in conventional prefixes

$$\log_2(1.724*10^{10}) = 34.0056684$$

hence , reach of TLB

$$2^{4.0056684}GB = 16.063\ GB$$

4 Question 4

given that the frame size is 1KB, the page offset bits in the physical address are 10 i.e. corresponding to the last 3 characters in the hexadecimal addresses given without the 2 most siginificant bits

Addresses are as follows:-

$$\begin{aligned}
page\ no\ at\ Port\ 1 &= concatenate(bin(0x4795B), 10) = a(say)\\
page\ no\ at\ Port\ 2 &= concatenate(bin(0x4795B), 10) = a\\
page\ no\ at\ Port\ 3 &= concatenate(bin(0x5795B), 10) = b(say)\\
page\ no\ at\ Port\ 3 &= concatenate(bin(0x4785B), 10) = c(say)
\end{aligned}$$

bin converts the hexadecimal to binary and its a simple post concatenation of the second argument after that to get the page number

we have 3 unique addresses out of 4, hence when using intra-cycle compaction we will need 3 unique accesses

5 Question 5

5.1 a)

Yes, the bits positioned at [30,29,28] are the same for the provided powers of 2 (-13 to -2); the bits are 0 1 1 respectively for the positions 30 29 28

5.2 b)

in that case , the corresponding bits will again be same for the given range of powers of 2 (1 to 11) and these bits will be 1 0 0

6 Question 6

Yes, it is possible to rewrite this code for improved cache performance

We perform 2 basic improvements:-

1.) do not build an intermediate array and loop only once substituting the value of a[i][j] in the second loop

2.) factorize the expression w.r.t c[i][j] so that it's accessed only once in each iteration and not twice

the renewed code is as follows

```
for(i=0;i<N;i++){
      for(j=0;j<N;j++){
            d[i][j] = c[i][j]*(1/b[i][j] + 1);
      }
}
```

Why this code is more cache friendly than the previous one :-

we're elminating the need to access a[i][j] which was being done twice in the original code

this leads to lesser cache misses because the working memory requirement has been reduced significantly (by 25%) and this will help reduce the numeber of capacity misses(obvious reason) and also the number of conflict misses because initially, when the array a was being accesed, it may have been displacing another cache block which contains the next accesses of b,c or d potentially due to the mappings which is probable given the information supplied in the question, this leads to that block staying in the cache and resulting in a successful hit the next time a corresponding location is accessed.

7 Question 7

	Value stored a location L in memory			
CPU activity	C1's cache	C2's cache	C3's cache	Memory
C1 reads L	5	-	-	5
C2 reads L	5	5	-	5
C2 writes 2 to L	-	2	-	2
C3 reads L	-	2	2	2
C3 writes 14 to L	-	-	14	14
C2 reads L	-	14	14	14
C1 writes 24 to L	24	-	-	24

logging cache states for a particular value if snooping protocols are used

8 Question 8

given that the page size is 1000B, the arrays a and c are mapped from the same page(because they consume 96 bytes each and taking note of their starting addresses, their mappings lie on the same page (5000 to 5999)) and arrays b and d (similar rationale)are mapped from another page. Hence, a simple operand reordering will improve the code for cache performance.

```
int main(){
      for(int i=0;i<24;i++){
            cout<<a[i] + c[i] + b[i] + d[i];
      }
}
```

9 Question 9

N and i are accessed repetitely and they show temporal locality

the array shows spatial locality in its accesses(we're coding in c/c++ which use row major storage in memory for arrays)

10 Question 10

The neccessary page tables have been loaded in the TLB so that those pages can be accessed without the compulsory misses in the next run of the program.

The same logic can be applied to cache tables, the relevant cache blocks will be preloaded in the cache in the next run further avoiding misses

This improves performance of the program in the second run and explains the person's observation