CS2323 Homework 2

CS18BTECH11001

October 16, 2019

This document is generated by LaTeX

Q1. Given, for a floating point number :

Total No. of Bits = T

No. of Exponent Bits = E

(a) ***Normal Number***

Smallest value :

Exponent = 0000....001

$\Rightarrow$ *actual exponent* $= 1 - (2^{E-1} - 1)$

$= 2 - 2^{E-1}$

Fraction = 000....00 $\Rightarrow$ *Significand* = 1.0

Smallest Value $= 1.0 \times 2^{2-2^{E-1}}$

$\therefore$ *Smallest value in normal mode* $= 2^{2-2^{E-1}}$

Largest value :

Exponent = 1111....110

$\Rightarrow$ *actual exponent* $= 2^E - 2 - (2^{E-1} - 1)$

$= 2^{E-1} - 1$

Fraction = 111....11 $\Rightarrow$ *Significand* $= 1 + 1 - 2^{E+1-T} = 2 - 2^{E+1-T}$

Largest Value $= (2 - 2^{E+1-T}) \times 2^{2^{E-1}-1}$

$\therefore$ *Largest value in normal mode* $= (2 - 2^{E+1-T}) \times 2^{2^{E-1}-1} \approx 2^{2^{E-1}}$

Denormal Number

Smallest value :

Exponent = 0000....000

$\Rightarrow$ *actual exponent* $= 0 - (2^{E-1} - 2)$

$= 2 - 2^{E-1}$

Fraction = 000....01 $\Rightarrow$ *Significand* $= 2^{-(T-E-1)}$

Smallest Value $= 2^{-(T-E-1)} \times 2^{2-2^{E-1}}$

$\therefore$ *Smallest value in Denormal mode* $= 2^{E+3-T-2^{E-1}}$

Largest value :

Exponent = 0000....000

$\Rightarrow$ *actual exponent* $= 0 - (2^{E-1} - 2)$

$= 2 - 2^{E-1}$

Fraction = 111....11 $\Rightarrow$ *Significand* $0 + 1 - 2^{E+1-T}$

Largest Value $= (1 - 2^{E+1-T}) \times 2^{2-2^{E-1}}$

$\therefore$ *Largest value in Denormal mode* $= (1 - 2^{E+1-T}) \times 2^{2-2^{E-1}} = 2^{2-2^{E-1}} - 2^{E+3-T-2^{E-1}}$

(b) ***FP16***

Total No. of Bits(T) = 16

No. of Exponent Bits(E) = 5

Smallest Normal Value $= 2^{2-2^{E-1}} = 2^{-14}$

Largest Normal Value $= (2 - 2^{E+1-T}) \times 2^{2^{E-1}-1} = 2^{16} - 2^5 \approx 2^{16}$

Smallest Denormal Value $= 2^{E+3-T-2^{E-1}} = 2^{-24}$

Largest Denormal Value $= 2^{2-2^{E-1}} - 2^{E+3-T-2^{E-1}} = 2^{-14} - 2^{-24}$

bfloat16

Total No. of Bits(T) = 16

No. of Exponent Bits(E) = 8

Smallest Normal Value $= 2^{2-2^{E-1}} = 2^{-126}$

Largest Normal Value $= (2 - 2^{E+1-T}) \times 2^{2^{E-1}-1} = 2^{128} - 2^{120} \approx 2^{128}$

Smallest Denormal Value $= 2^{E+3-T-2^{E-1}} = 2^{-133}$

Largest Denormal Value $= 2^{2-2^{E-1}} - 2^{E+3-T-2^{E-1}} = 2^{-126} - 2^{-133}$

(c) *Second Smallest Normal Value :*

Exponent = 0000....001

$\Rightarrow$ *actual exponent* $= 1 - (2^{E-1} - 1)$

$= 2 - 2^{E-1}$

Fraction = 000....01 $\Rightarrow$ *Significand* $= 1 + 2^{-(T-E-1)}$

Second Smallest Value $= (1 + 2^{-(T-E-1)}) \times 2^{2-2^{E-1}} = 2^{2-2^{E-1}} + 2^{E+3-T-2^{E-1}}$

$\therefore$ *Second Smallest Normal Value* $= 2^{2-2^{E-1}} + 2^{E+3-T-2^{E-1}}$

FP16

Smallest Normal Value $= 2^{2-2^{E-1}} = 2^{-14}$

Second Smallest Normal Value $= 2^{2-2^{E-1}} + 2^{E+3-T-2^{E-1}} = 2^{-14} + 2^{-24}$

So, Difference $= 2^{-14} + 2^{-24} - 2^{-14} = 2^{-24}$

$\therefore$ *Difference* $= 2^{-24}$

bfloat16

Smallest Normal Value $= 2^{2-2^{E-1}} = 2^{-126}$

Second Smallest Normal Value $= 2^{2-2^{E-1}} + 2^{E+3-T-2^{E-1}} = 2^{-126} + 2^{-133}$

So, Difference $= 2^{-126} + 2^{-133} - 2^{-126} = 2^{-133}$

$\boxed{\therefore\ Difference = 2^{-133}}$

(d) ***FP16***

Pros :

- *FP16 has high precision than bfloat16 due to more mantissa bits*

Cons :

- *FP16 has low range compared to bfloat16 due to less exponent bits*
- *Conversion between FP16 and FP32 is difficult due to different no. of exponent bits as FP32*

bfloat16

Pros :

- *bfloat16 has high range compared to FP16 due to more mantissa bits*
- *Conversion between bfloat16 and FP32 is simple due same no. of exponent bits as FP32*

Cons :

- *bfloat16 has much low precision near 1 than FP16*

(e) *The relative spacing between two consecutive numbers is more in bfloat when compared to FP16 due to less no. of mantissa bits. So, the Range of bfloat16(2^{129}) is much larger compared with that of the Range of FP16(2^{17}).*

Q2. Given,

No.of bits in Virtual Address $= 48$

No. of entries $= 64$

Physical memory $= 2GB = 2 \times 1024 \times 1024 \times 1024 = 2^{31}B$

$\Rightarrow$ *No. of bits in physical Address* $= \log_2^{2^{31}} = 31$

Page Size $= 2KB = 2 \times 1024 = 2^{11}$

$\Rightarrow$ *No. of page offset bits* $= \log_2^{2^{11}} = 11$

So,No. of page number bits $= 48 - 11 = 37$

No. of frame number bits $= 31 - 11 = 20$

w.k.t TLB Size $= No.\ of\ Entries \times (No.\ of\ page\ number\ bits + No.\ of\ frame\ number\ bits)$

$= 64 \times (37 + 20)$

$= 64 \times 57$

$= 3648Bits$

$\boxed{\therefore TLB\ Size = 3648Bits}$

Q3. TLB Coverage(or Reach) $= \Sigma\ (pagesize) \times (Entries)$

$= 4KB \times 128 + 2MB \times 32 + 2GB \times 8$

$= 512KB\ 64MB\ 16GB$

$= 16,843,264KB$

$\boxed{\therefore\ TLB\ Coverage = 16,843,264KB}$

Q4. *Given,*

Frame Size $= 1KB = 1 \times 1024 = 2^{10}$

$\Rightarrow$ *No. of Frame offset bits = No. of page offset bits* $= \log_2^{2^{10}} = 10$

No. of bits in the Given Addresses $= 8 \times 4 = 32$

$\Rightarrow$ *No. of page number bits* $= 32 - 10 = 22$

So, for the intra-cycle compaction we have to consider the first 22 bits as Corresponding VPNs

Hexadecimal VA	***Binary VA***	***Binary VPN***	***VPN after compaction***
0x4795BA21	*01000111100101011011101000100001*	*0100011110010101101110*	*0100011110010101101110*
0x4795BB21	*01000111100101011011101100100001*	*0100011110010101101110*	————
0x5795BA21	*01010111100101011011101000100001*	*0101011110010101101110*	*0101011110010101101110*
0x4785BA21	*01000111100001011011101000100001*	*0100011110000101101110*	*0100011110000101101110*

$\therefore$ *Total No. of unique accesses sent to* $TLB = 3$

Q5. *(a) Yes, The bits* $[30, 29, 28]$ *of all the weights in the range* $[2^{-13} : 2^{-2}]$ *are same and equal to* 0 1 1.

(b) The value of bits $[30, 29, 28]$ *of all the weights in the range* $[2^1 : 2^{11}]$ *are same and equal to* 1 0 0.

Q6. *Yes, It's possible to reduce the no. of L1 cache misses*

Code :

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

Q7. *Invalidating Snooping Protocol :*

Given, In the memory location 'L' value of 5 is stored

		Value stored at location L in			
CPU Activity	*activity on bus*	*C1's Cache*	*C2's Cache*	*C3's Cache*	*Memory*
C1 reads L	*Cache miss for L*	*5*			*5*
C2 reads L	*Cache miss for L*	*5*	*5*		*5*
C2 writes 2 to L	*Invalidate for L*		*2*		*2*
C3 reads L	*Cache miss for L*		*2*	*2*	*2*
C3 writes 14 to L	*Invalidate for L*			*14*	*14*
C2 reads L	*Cache miss for L*		*14*	*14*	*14*
C1 writes 24 to L	*Invalidate for L*	*24*			*24*

Q8. *Yes, It's possible to improve the TLB efficiency*

Code :

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

Q9. *Variables showing Temporal locality : i,N*

Variables showing Spacial locality : array[N]

Q10. For the first time when the Person1 runs the program it take more time because the data has to be fetched to the cache which includes compulsory misses. For the Second time the as data is already stored in the cache the compulsory misses will be reduced and time is reduced.