(1, 10) (11, 10), (4, 12) (4, 15).

COMPUTER ARCHITECHTURE

Sai Harsha - K
CS17BTECH11036

1) Single precision floating number

$\underbrace{0}_{\text{signed}}\underbrace{01111101}_{\text{exponent}}\ \underbrace{01010101010101010101010}_{\text{fraction}}$

decimal representation $(x) = (-1)^s \times (1 + \text{Fraction}) \times 2^{(\text{exponent} - \text{bias})}$

here, bias = 127.

$S = 0.$

$E = (01111101)_2 = (125)_{10}.$

$1 + \text{Frac} = \text{significand} = 1.\underbrace{0101....010}$

$(x) = (-1)^0 + (1.0101\overbrace{\ldots 010}^{22\text{ bits}}) \times 2^{-2} = \underline{0.333333}\ldots$

~~2) Decimal representation = 5.6677.~~

2) Decimal : 5.6677.

Single precision floating point: 01000000101101010101110111001100

Value in Decimal : 5.667699813843.

difference: 1.8615722656 25E -7.

---

double : 0100000000010110101010111011100110001100011111100010100000100100.

value in decimal : 5.6676999999999996

difference : 3.97790393202570E -17.

~~2) Decimal → 5.6677.~~

~~In single precision floating point : 0 10000001 01101010101110111001100~~

~~This value in decimal : 5.66769999999999996~~

~~difference :~~

3) v.ld vr1, [0x1000].

semantics: vr1 ← ([0x1000], [0x1004], [0x1008], [0x100C]).

4)

Request	open	closed
(4, 15)	miss	miss
(4, 6)	conflict	miss.
(6, 150)	conflict	miss
(7, 150)	hit	miss
(3, 9)	conflict	miss
(4, 9)	hit	miss
(4, 12)	conflict	miss
(4, 156)	conflict	miss
(5, 10)	conflict	miss
(6, 10)	hit	miss
(11, 10)	hit	miss

open-row : ~~Total misses = 7.~~ Misses : 1; conflicts : 6.

closed-row : Total misses = 11.

Scanned by CamScanner

5) (4,6), (3,9), (4,9), (5,10), (6,10), (11,10), (4,12), (4,15), (4,150), (6,150), (7,150)

6) It should be high in this case, number of hits will increase since, with high spatial locality, ~~same~~ application is likely to access the same row avoiding any ~~misses~~ conflicts.

7) There is no definite Yes/No.

→ Increases energy consumption: making predictions on biased branches. predictions are not taken.

→ Decreases energy consumption: predictions are taken

8) weight vector $= [10, -4, 19, -2, 2]$ → Bias.

BHR $= [1, -1, 1, 1]$

$\Rightarrow (10\times1) + (4) + (19)(-2) + 2 = 33.$

since $33 > 0$.

⇒ Taken.

Given outcome is Taken so we update weight vector ~~to~~ as:

weight vector $= [11, -5, 20, -1, 3]$.

BHR $= [1, -1, 1, 1]$

$\Rightarrow 11 + 5 + 20 + 1 + 3 = 38 \Rightarrow$ Taken

Scanned by CamScanner

9) It increases soft error rate.
voltage scaling technique allows lower energy particles to flip bits unlike ~~earlier~~ previous case.

10) ld r1, 8[r2]
st r1, 16[r3]

11) ~~sub r1, r2, r3~~
~~sub add r2, r1, r3~~. WAR hazard : add r1, r2, r3.
add r3, r2, r4.

12) [1] beq .b1.
[2] mov r1, 0
[3] sub r3, r4, r2.
. . . . .
. . . . .
.b1:
[100] add r4, r1, r3.

13) For the given sequence of instructions,
there is load-use hazard.
Hence, we must add bubble.

14)

IF	1	2	3						
OF		1	2	3	3	3			
EX			1	2	*	*	3		
MA				1	2	*	*	3	
RW					1	2	*	*	3.

* implies pipeline bubble.

15)

Request	Time of Arrival (ns)	Open-row	Closed-row
X_0	0	40	40
Y_0	10	100	100
X_1	100	160	160
X_2	200	220	240
Y_0	250	310	300
X_3	300	370	360

Row buffer hit = 20 ns, Row-buffer conflict = ~~40~~ 60 ns, Row-buffer miss = 40 ns.

16)

strongly coupled MP	vs	loosely coupled MP
→ Programs run in parallel sharing code, data, files and network connections on a multiprocessor.		→ Programs running are unrelated, run on parallel on a multiprocessor
→ ~~Own memory, which is not shared with other processes~~		→ ~~usually share mem~~
→ usually share memory as they use same data.		→ Own memory, which is not shared with other processes.

17) Total chips in DRAM $= \dfrac{8\text{Mb} \times 4}{1\text{Mb}} = \underline{\underline{32}}$

4 chips can be refreshed in parallel.

$\Rightarrow \dfrac{32}{4} = 8$ steps required to refresh all chips

Total time taken $= 8 \times \underbrace{(1000 \times 1000)}_{\text{Total number of cells.}} \times 10^{-9} \times 100\text{s} = \underline{\underline{0.8\text{s}}}$.

Scanned by CamScanner

18) Simple Pipelining

fetch decode add
fetch decode mul
fetch decode add
fetch decode sub
fetch decode ld

superscalar execution

fetch decode add.
fetch decode mul
fetch decode add
fetch decode sub
fetch decode ld.

Out of Order execution

fetch decode add
fetch decode sub
fetch decode __ mul
fetch decode __ ld
fetch decode __ __ add.
fetch decode

19) We should use ~~course~~ coarse grain multithreading strategies

20) v.cmp vr1, 50

v.gt.mul vr1, vr1, 50

21) Number of ranks per channel = 4

64 bit wide data ⇒ 8 chips per rank.

Total capacity of DRAM = 8 × 16 Gb × 4. ; size of one chip = 16 Gb.

= 64 Gb.

22) A C Z V K L Z V C M R B A K

Reuse distance = 6. [all access]

Reuse distance = 4 [distinct access]

23) ~~for (k=0; k<R; k~~

```
for (row = 0; row < R; row = row + B )
  for (a = row; a < min (R, row + B); a ++ )
    for (col = 0; col < C; col ++ )
      for (to = 0; to < M; to ++ )
        for (ti = 0; ti < N; ti ++ )
          for (i = 0; i < k; i ++ )
            for (j = 0; j < k; j ++ )
            { Output_fmaps [to][row][col] += Weights[to][ti][i][j] *
                Input_fmaps [ti][s*row + i][s*col + j]
            }
```

Scanned by CamScanner

24) a) → N = 1 (or) N ≥ 100

say biased implies 99% taken or not not taken.

→ Not conditions on array [ ].

b) → No condition on N

→ More than half of array should be divisible by 20

(or)

More than half of array should be not divisible by 20.

25) Temporal multi-bit errors:

Use "scrubbing" => ~~Read data~~ Uses ECC to correct data and then write that data.

→ we need to correct data before it exceeds the capacity of ~~correction~~ corrigible errors.

Spatial multi-bit errors

Use "Bit interleaving":

~~Playing with~~

Choose arrangement of blocks so that maximum ECC can be applied where each sub block has error not exceeding capability of ECC.