CSEC 20th – Past Year Paper Solution 2018-2019 Sem1

CE/CZ1006 – Computer Organisation and Architecture

Question 1

1)

a)

i)	R0: 0x999	SR: 0x004
ii)	R0: 0x000	SR: 0x002
iii)	R0: 0xF3E	SR: 0x004
iv)	R0: 0x750	SR: 0x009
v)	R0: 0xE65	SR: 0x005

b) **1st_Check**: R0: 0x00A, R1: 0x000, R2: 0x102, R3: 0xF32, SR: 0x003
2nd_Check: R0: 0x922, R1: 0x102, R2: 0x00F, R3: 0x102, SR: 0x000

c) 1. At 0x001, **MOV RO, #0** should be replaced with **MOVS RO, #0**
2. At 0x005, **ADD R0, #1** should be replaced with **INC R0**
3. At 0x007, **SUB R1, #1** should be replaced with **DEC R1**

Question 2

2)

a)
```
I1: PSH [0x100]
I2: PSH [0x101]
I3: PSH #0x102
I4: MOV R0, [SP+6]
I5: MOV R1, [SP+5]
I6: MOV R2, [SP+4]
I7: MOV [R2+0], R0
I8: RET
```

b) Vx = 0x005
Vy = 0x003
Vz = 0x028

c) Vx * 2^Vy = Vz

d)
```
If(Vy<0){

      Vy = Vy * -1;

  }
```

Question 3

3)

a)

i) NAND Memory. It takes up little space, is non-volatile, is relatively resistant to drop impacts, which a smartphone may suffer during its course of usage and has an acceptable cost-per-bit ratio.

ii) Registers use SRAM as registers require speed, can be volatile, and are small, with size not being a disadvantage. SRAM fulfils all these requirements.

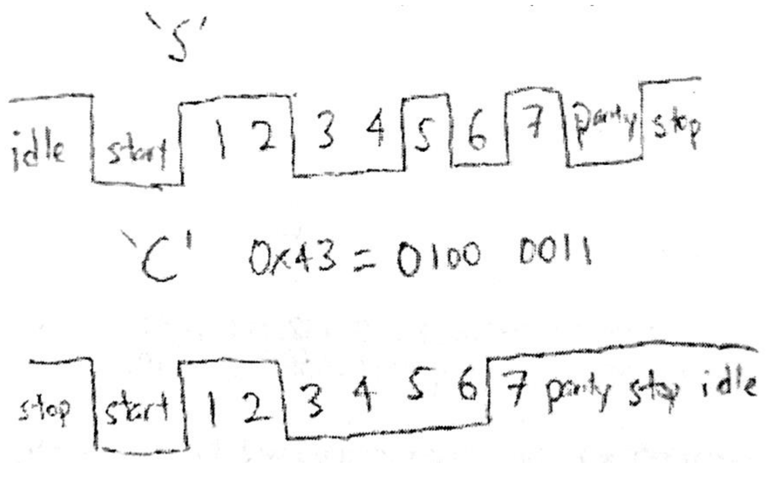
iii) Cache is faster than internal RAM.

b)

i) There could be an issue of signal skew in the graphics card. This can be mitigated by redrawing the shorter PCB trace by making it take a longer, more winding route, and therefore have equal lengths with the previously longer PCB trace.

ii) Add more data lines such that each GDDR5 chip has 256 data lines instead of 32 since each line has a maximum data rate of 1GBytes/s. The assumption is that there is enough space to do so and doing so would not increase signal skew or crosstalk to the extent that it degrades performance to the point where 256GBytes/s transfer cannot be achieved.

c) 1 start bit, 7 data bits (LSB first), 1 even parity bit, 1 stop bit



d) Number of bytes per track = 256 *512= 131072 bytes

Number of bytes of data to be transferred = $(262*2^{10})$ = 268288 bytes

268288/131072 = 2.046

Hence, the HDD needs to read the data from 3 tracks to transfer data, assuming the data is kept sequentially.

RPS = 12000/60 = 200 per second

Access time $T_A = T_s + T_R$ = 3*(5 ms + (0.5/200)) = 0.0225s

Transfer time T_t = 2*(131072/(200*256*512)) + 6144/(200*256*512) +0.0225= 0.0327s

Question 4

4)

a) Write allocate => fetch on write. A write miss will cause the data block at the write-miss address to be loaded to the cache.
Write-no-allocate => a write miss will not cause the data block to be loaded to the cache. Data write is done directly to its location in main memory.
Either policy will be used when a write-miss occurs.
Write-no-allocate is better for a system that performs writes to many random addresses as it takes additional time to load the memory into the cache, and the randomness of the process means that it is unlikely that the data will be required by the system during this process.

b) (0.8*5)+(0.2*0.9*15)+(0.2*0.1*115) = 9ns

c) Rounding errors when adding two numbers with significant but not overwhelming differences in exponents generally causes truncation of lower-order data bits that results in a loss of accuracy.

Example 1: 1.23e01+4.69e00 =

```
   1 . 2 3 e + 0 1
 + 4 . 6 9 e + 0 0
 -----------------
   1 . 2 3 e + 0 1
 + 0 . 4 6 e + 0 1
 -----------------
   1 . 6 9 e + 0 1
```

But there is a loss of precision as the ‘9’ digit 4.69e00 is ignored. To mitigate this issue, addition should be performed with a **guard bit** such that:

```
   1 . 2 3 0 e + 0 1
 + 4 . 6 9 0 e + 0 0
 -------------------
   1 . 2 3 0 e + 0 1
 + 0 . 4 6 9 e + 0 1
 -------------------
   1 . 6 9 9 e + 0 1 ≈ 1 . 7 0 e + 0 1
```

This ends up with a result that is more accurate.

The second possibility, even with guard bits, is when multiple small values are added to a much larger number, such that:

Example 2: 1.23e03+3.00e00 + 3.00e00 =

```
  1.230e+03
+ 3.000e+00
  1.230e+03
+ 0.003e+03
  1.233e+03≈1.23e+03
```

```
  1.230e+03
+ 3.000e+00
  1.230e+03
+ 0.003e+03
  1.233e+03≈1.23e+03
```

This results in a loss of accuracy as the true result, before rounding up, would be 1.236e03, which is closer to 1.24e03 than 1.23e03. Therefore, smaller values should be added to each other before greater values such that:

```
  3.000e+00
+ 3.000e+00
  6.000e+00≈6.00e+00

  6.000e+00
+ 1.230e+03
  0.006e+03
+ 1.230e+03
  1.236e+03≈1.24e+03
```

d) F = Fetch, D = Decode, E= Execute, S = Store

```
I1    [F ][D ][E ][S ](MOV R0, #200)
I2        [F ][D ][E ][S ](MOV AR, #10)
I3            [F ][D ][E ](Loop JDAR Loop)(Data Conflict)
```

There is a data conflict the first time I3 runs as I2 has not finished storing data in AR, and I3 is dependent on the result of I2. To resolve this issue, I2 should be executed before I1.

```
I5    [F ][D ][E ][S ](INC R0)
I6        [F ][D ][E ][S ](MOV R2, R0)(Data Conflict)
I7            [F ][D ][E ][S ](MOV R3, R1)
```

There is a data conflict at I6 because I5's instruction increases R0 by 1 but has not finished its storage pipeline stage before I6 reaches its execution stage. To fix this issue, I7's instruction can be moved in front of I6 and I7's result is not dependent on I6.

The final set of instructions should be:

MOV AR, #10 (I2)

MOV R0, #200 (I1)

Loop JDAR Loop (I3)

```
ADD R1, [R0] (I4)

INC R0 (I5)

MOV R3, R1 (I7)

MOV R2, R0 (I6)
```

==End of Answers==

Solver: Tan Song Xin, Alastair

Email Address: tans0380@e.ntu.edu.sg