**21st CSEC – Past Year Paper Solution (2017 – 2018 Semester 2)**
**CE/CZ 3001 – Advance Computer Architecture**

1 ai) M1 clock rate = 80MHz
M2 clock rate = 100MHz

M1 avg. CPI = $1 \times 0.6 + 2 \times 0.3 + 4 \times 0.1 = 1.6$
M2 avg. CPI = $2 \times 0.6 + 3 \times 0.3 + 4 \times 0.1 = 2.5$

Speedup of M1 over M2 = $\frac{Time\ taken\ to\ complete\ a\ set\ of\ instructions\ for\ M2}{Time\ taken\ to\ complete\ a\ set\ of\ instructions\ for\ M1}$

$= \frac{2.5 \times \frac{1}{100 \times 10^6}}{1.6 \times \frac{1}{80 \times 10^6}}$

$= 1.25$

1 aii) Applying Gustafon's law,

Speedup of new M1 over old M1 = $\frac{Work\ done\ for\ new\ M1\ in\ 1\ clock\ cycle}{Work\ done\ for\ old\ M1\ in\ 1\ clock\ cycle}$

$= \frac{\frac{1}{1.6} \times 0.25 + \frac{1}{1.6} \times 0.75 \times 10}{\frac{1}{1.6}}$

$= 7.75$

1 aiii) Since maximum operation frequency $f_{max} \propto \frac{[V - V_{th}]}{V}$,

Assuming $V_{th}$ is negligible, $f_{at\ 4V} = f_{at\ 5V} \times \frac{V_{new}}{V_{old}}$

$= 100 \times \frac{4}{5} = 80MHz$

Dynamic power consumption = $ACV^2f$

Percentage change in Dynamic Power Consumption = $\frac{ACV^2f_{new} - ACV^2f_{old}}{ACV^2f_{old}} \times 100\%$

$= \frac{abs(4 \times 4 \times 80 - 5 \times 5 \times 100)}{5 \times 5 \times 100} \times 100\%$

$= 48.8\%$

Static power consumption = $VI_{leak}$

Percentage change in Dynamic Power Consumption = $\frac{abs(V_{new}I_{leak} - V_{old}I_{leak})}{V_{old}I_{leak}} \times 100\%$

$= \frac{abs(4 - 5)}{5} \times 100\%$

$= 20\%$

1 b)
```
        ADDI    $t0,    $zero,  0x003F
        ADD     $t1,    $zero,  $zero
        LW      $a0,    0 ($s1)
LOOP    LW      $a1,    4 ($s1)
        JAL     min-2
        ADDI    $t1,    $t1,    0x0001
        ADD     $s1,    $s1,    0x0004
        BNE     $t0,    $t1,    LOOP
        LUI     $t2,    0x000F
        ORI     $t2,    0x0204
        SW      $a0,    0 ($t2)
        JR      $ra
```

2 ai)

| Clock | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| I1 | IF | ID | EX | M | WB | | | | | | | | |
| I2 | | IF | ID | EX | M | WB | | | | | | | |
| I3 | | | IF | ID | EX | M | WB | | | | | | |
| I4 | | | | IF | ID | EX | M | WB | | | | | |
| I5 | | | | | IF | S | ID | EX | M | WB | | | |
| I6 | | | | | | | IF | ID | EX | M | WB | | |
| I7 | | | | | | | | IF | ID | EX | M | WB | |
| I8 | | | | | | | | | IF | ID | EX | M | WB |

Assume 2 branch cycles after branching.

1 stall cycle (between I4 and I5 is required) + 2 stall cycles (after branch) = 3 stall cycles

2 aii) Since PC is only updated after M, IF for next instruction after branch can only start after 3 stall cycles

$$Steady\ state\ CPI = \frac{(No.of\ instructions + no.of\ stalls)}{no.of\ instructions}$$

$$= \frac{8+1+3}{8} = 1.5$$

2 b) BEQ $zero, $zero, offset

Branch is an 'I' Type instruction. Hence the immediate (offset) value can only contain 16 bits of data. However, the most significant bit is a signed bit.

Hence Min. value of offset is 8000 and Max. value of offset is 7FFF. This value is left shifted twice before summing with the current PC. i.e. old PC + 4 + offset << 2

Min. achievable address = 0x10EE FF10

Max. achievable address = 0x10F2 FF0C

J target

Jump is a 'J' type instruction. Hence the target value contains 26 bits.

Smallest value = 0x000 0000
Largest value = 0x3FF FFFF

Jump address is calculated by taking the first 4 bits of the current pc, concatenate it with the target value after left shifting it twice. i.e. (old PC + 4) [31:28](target<<2)[27:0]

Min. achievable address = 0x1000 0000
Max. achievable address = 0x1FFF FFFC

2 c) Note: It doesn't matter which state you start in, the 8 Ts will set you in the Predict Taken, 11 (top left) state.

| ACTUAL | N | N | T | N | T | T | N | N | T | N |
|---|---|---|---|---|---|---|---|---|---|---|
| PREDICT | T | T | N | N | N | N | T | T | N | N |

Prediction Accuracy = $\frac{2}{10} \times 100\% = 20\%$

3 a) 1. Single Instruction Single Data stream
2. Single Instruction Multiple Data Stream
3. Multiple Instruction Single Data stream
4. Multiple Instruction Multiple Data Stream

Single Instruction Multiple Data Stream best supports data level parallelism

3 b) For N-way set-associative cache, # set = Total cache size / (Block size x N)

For 4-way set-associative cache, # set = 64KB / (16 x 4) = 2^16/(2^4 x 2^2) = 2^10.

4-way – set index = $\log_2$ ( 2^10 )= 10 bits

Block offset = $\log_2$(2^4 ) = 4 bits

Tag = 32 (address space) - 10 - 4 = 18 bits

3 ci) Each virtual page is 1024 bytes large, starting from 0 bytes, these are the bytes assigned to each virtual page:

| Page | Bytes |
|---|---|
| 0 | 0 – 1023 |
| 1 | 1024 – 2047 |
| 2 | 2048 – 3071 |
| 3 | 3072 – 4095 |
| 4 | 4096 – 5119 |
| 5 | 5120 – 6143 |
| 6 | 6144 – 7167 |
| 7 | 7168 – 8192 |

Virtual pages 2, 3, 5, 7 will trigger page faults.

3 cii) No. of bits required to encode Virtual pages = 3bits
No. of bits required to encode Physical pages = 2bits

$1023_{10} = 0\ 0011\ 1111\ 1111_2$
$Physical\ page = 1111\ 1111\ 1111_2$
$= 4095_{10}$

$1024_{10} = 0\ 0100\ 0000\ 0000$
$Physical\ page = 0100\ 0000\ \ 0000_2$
$= 1024_{10}$

One extra '0' bit is added as we already know that we need 3 bits to encode the virtual pages and that the byte 1023 lies in the 0th page. You can check that this logic is sound by observing the first 3 bits of the 1024th byte is 001, the 1st virtual page

4 a) Note: This question seems tough at first but is actually quite easy in the sense that all the answers are already laid out, you can even verify that your method is correct by calculating out the values that were already given to you. The tough impression may come from the fact that there are a lot of figures and words to go through.

$Avg.Memory\ Access\ Time = \ Time\ for\ hit + Miss\ rate\ \times Miss\ Penalty$

For X,

$1.93 = 1 + X \times 88$

$X = \frac{1.93-1}{88}$

$X = 1.06\%\ (3\ s.f.)$

For Y,

$Y = 1 + 2.64\% \times 88$

$Y =$ 3.32 clock cycles

For Z,

$Z = 1 + 0.51\% \times 88$

$Z = 1.45\ (3\ s.f.)$

4 b) The 64B block size has the smallest AMAT for all 4 cache sizes.

For the 4KB, 16KB and 64KB cache size, the 64KB block size has the lowest miss rate. For the 256KB cache size, the 256B block size has the lowest miss rate.

4 c) 1. Increase in block size allows more exploitation of spatial locality, leading to potentially less misses.

2. Too much of an increase in block size results in fewer placement choices, leading to more conflict misses and higher miss rates.

4 d) Average Memory Access Time. AMAT considers both hit time and miss time to the cache whereas miss rate only considers miss rates of the cache.

One example to show that AMAT is superior would be to consider two different caches with similar miss rates, but drastically different hit times. Using the miss rate metric, we would rate both caches the same. Using the AMAT metric, a cache with a lower hit time or lower miss penalty will outperform a cache with a higher respective time, assuming all other variables are the same.