Problem No. 1

(a) Number of banks = 8

Therefore, number of bits needed to address the **Bank** field = \( \log_2(8) = 3 \)

Capacity of each bank = \( \frac{4 \text{Gbytes}}{8} = 2^{32} / 8 = 2^3 \) bytes

Capacity of each row = \( 4 \text{Kbytes} = 2^{12} \) bytes

Therefore, number of rows per bank = \( 2^{29} / 2^{12} = 2^{17} \)

Hence, the number of bits needed to address the **Row** field = 17

For every address, the most significant 17 bits provide the row number and the next 3 bits provide the bank number.

<table>
<thead>
<tr>
<th>Address (hex)</th>
<th>Bank number</th>
<th>Row number (decimal)</th>
<th>Row buffer hit/miss</th>
<th>Comments</th>
<th>Latency</th>
</tr>
</thead>
<tbody>
<tr>
<td>46A22000</td>
<td>2</td>
<td>36164</td>
<td>Miss</td>
<td>Row#36164 opened in bank#2</td>
<td>( t_{\text{RCD}} + t_{\text{CL}} + t_{\text{BURST}} )</td>
</tr>
<tr>
<td>869D5440</td>
<td>5</td>
<td>68922</td>
<td>Miss</td>
<td>Row#68922 opened in bank#5</td>
<td>( t_{\text{RCD}} + t_{\text{CL}} + t_{\text{BURST}} )</td>
</tr>
<tr>
<td>A43C2000</td>
<td>2</td>
<td>84088</td>
<td>Miss</td>
<td>Row#84088 opened in bank#2</td>
<td>( t_{\text{RP}} + t_{\text{RCD}} + t_{\text{CL}} + t_{\text{BURST}} )</td>
</tr>
<tr>
<td>59320680</td>
<td>0</td>
<td>45668</td>
<td>Miss</td>
<td>Row#45668 opened in bank#0</td>
<td>( t_{\text{RCD}} + t_{\text{CL}} + t_{\text{BURST}} )</td>
</tr>
<tr>
<td>46A22040</td>
<td>2</td>
<td>36164</td>
<td>Miss</td>
<td>Row#36164 opened in bank#2</td>
<td>( t_{\text{RP}} + t_{\text{RCD}} + t_{\text{CL}} + t_{\text{BURST}} )</td>
</tr>
<tr>
<td>593204C0</td>
<td>0</td>
<td>45668</td>
<td>Hit</td>
<td>Hit in open row in bank#0</td>
<td>( t_{\text{CL}} + t_{\text{BURST}} )</td>
</tr>
<tr>
<td>869D5000</td>
<td>5</td>
<td>68922</td>
<td>Hit</td>
<td>Hit in open row in bank#5</td>
<td>( t_{\text{CL}} + t_{\text{BURST}} )</td>
</tr>
</tbody>
</table>

(b) DRAM clock speed = 667 MHz

Therefore 1 DRAM clock cycle = \( \frac{1}{667 \text{MHz}} = 1.5 \) nsec

Since burst length = 8 and there are two data transfers per cycle in DDR, \( t_{\text{BURST}} = \frac{8}{2} = 4 \) cycles

Row hit latency (requests # 6 and 7) = \( t_{\text{CL}} + t_{\text{BURST}} = 5 \) cycles + 4 cycles = 9 cycles

Row miss latency with no row currently open (requests # 1, 2 and 4) = \( t_{\text{RCD}} + t_{\text{CL}} + t_{\text{BURST}} = 10 + 5 + 4 = 19 \) cycles

Row miss latency when there is an open row which first needs to be closed (requests # 3 and 5) = \( t_{\text{RP}} + t_{\text{RCD}} + t_{\text{CL}} + t_{\text{BURST}} = 10 + 10 + 5 + 4 = 29 \) cycles

Average DRAM latency = \( (9 \times (2/7)) + (19 \times (3/7)) + (29 \times (2/7)) = 19 \) cycles = \( 19 \times 1.5 \) nsec = 28.5 nsec

Note: The arrival time data for each request and the \( t_{\text{RAS}} \) values were supplied to you, so that you could account for queuing delays (if there were any). For the given arrival times, you can validate that whenever a request arrives at a bank, the previous request has already been completed (zero queuing delays). For example, the first request to bank#2 completes at 35ns, whereas the second request to the same bank arrives at 60ns. If the second request had arrived earlier, then it may have had to wait for the previous request to be completed, which could have resulted in additional latency.
**Problem No. 2**

(a) Loop-A is expected to have more instruction cache hits. This is because loop-A has more iterations and thus the same instructions keep on getting accessed over and over again. Furthermore, loop-A has fewer instructions in each iteration, causing the instruction footprint to easily fit in a small cache.

(b) Increasing the cache block size results in more data being transferred from the main memory to the cache on every cache miss.

Advantage: This can result in a higher cache hit ratio, if the application exhibits high spatial locality.

Disadvantages: (i) This can increase the cache miss latency and put more bandwidth pressure on the next level cache (or memory), (ii) It can result in fragmentation and wastage of capacity, if the program does not exhibit high spatial locality.

**Problem No. 3**

(a) Block size = 64 bytes = 2⁶ bytes. Therefore, Number of bits in the Byte Select field = 6

 Cache size = 32K-byte = 2¹⁵ bytes

 Total number of cache blocks = Cache size / Block size = 2¹⁵ / 2⁶ = 2⁹ = 512

 Number of cache blocks per set = 1 (Direct-mapped cache)

 Number of sets = 2⁹ / 1 = 2⁹. Therefore, Number of bits in the Index field = 9

 Number of bits in the Tag field = 32 - 6 - 9 = 17

(b) Each cache block requires the following SRAM bits: 17 tag bits, 1 valid bit, 1 dirty bit and 512 data bits

 Therefore, number of SRAM bits needed for each block = 17 + 1 + 1 + 512 = 531

 Total SRAM bits in the cache = Total number of blocks * SRAM bits needed for each block

 = 512 * 531

 = 271,872 bits

**Problem No. 4**

(a) Block size = 128 bytes = 2⁷ bytes. Therefore, Number of bits in the Byte Select field = 7

 Cache size = 4M-byte = 2²² bytes

 Total number of cache blocks = Cache size / Block size = 2²² / 2⁷ = 2¹⁵ = 32768

 Number of cache blocks per set = 16 = 2⁴

 Number of sets = 2¹⁵ / 2⁴ = 2¹¹. Therefore, Number of bits in the Index field = 11

 Number of bits in the Tag field = 32 - 7 - 11 = 14

(b) Each cache block requires the following SRAM bits: 14 tag bits, 1 valid bit, 1 dirty bit and 1024 data bits

 Therefore, number of SRAM bits needed for each block = 14 + 1 + 1 + 1024 = 1040

 Total SRAM bits in the cache = Total number of blocks * SRAM bits needed for each block

 = 32,768 * 1040

 = 34,078,720 bits
Problem No. 5

(a) LRU policy

Initial contents of the set: empty
Access#1 (A): miss, New set contents: A
Access#2 (A): hit
Access#3 (B): miss, New set contents: A, B; LRU = A
Access#4 (C) miss, C replaces A, New set contents: B, C; LRU = B
Access#5 (B): hit, C becomes LRU
Access#6 (C): hit, B becomes LRU
Access#7 (A): miss, A replaces B, New set contents: C, A; LRU = C
Number of hits = 3

(b) The following access sequence results in 2 hits for LFU but only 1 hit for LRU:
A, A, B, C, A

LFW policy

Access#1 (A): miss, New set contents: A
Access#2 (A): hit
Access#3 (B): miss, New set contents: A, B; LFU = B
Access#4 (C) miss, C replaces B, New set contents: A, C; LFU = C
Access#5 (B): miss, B replaces C, New set contents: A, B; LFU = B
Access#6 (C): miss, C replaces B, New set contents: A, C; LFU = C
Access#7 (A): hit
Number of hits = 2