ECE341 Homework No. 8 Solution

Problem No. 1

For each 16M x 8 memory chip:

Word size = 8 bits
Memory capacity = 16M * 8 = $2^{24} * 2^3 = 2^{27}$ bits

For the entire 64M x 128 memory:

Word size = 128 bits
Total memory capacity = 64M * 128 = $2^{26} * 2^7 = 2^{33}$ bits

Number of memory chips needed = Total memory size / Size of each chip = $2^{33}/2^{27} = 2^6 = 64$
Number of memory chips needed per row = Word size for entire memory / Word size for each chip = 128/8 = 16
Number of rows needed = Total number of memory chips / Memory chips per row = 64 / 16 = 4

The required structure is essentially the same as in Figure 8.10, except that each of the **four** rows has **sixteen** 16M x 8 chips. Address lines $A_{19\text{-}0}$ should be connected to all chips. Address lines $A_{21\text{-}20}$ should be connected to a 2-bit decoder to select one of the 4 rows.

Problem No. 2

(a) Temporal locality refers to the program property that recently accessed data/instructions are likely to be accessed again very soon. Spatial locality refers to the property that if one address is accessed by a program, then the data/instructions in the spatially adjacent (or nearby) addresses are likely to be accessed as well.

(b) 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.

(c) 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.
Duasvantages: (i) This can increase the cache miss latency, (ii) It can result in fragmentation and wastage of capacity, if the program does not exhibit high spatial locality.

Problem No. 3

Block size = 64 bytes = 2^6 bytes = 2^6 words (since 1 word = 1 byte)
Therefore, **Number of bits in the *Word* field = 6**
Cache size = 64K-byte = 2^{16} bytes
Number of cache blocks = Cache size / Block size = $2^{16}/2^6 = 2^{10}$

Therefore, **Number of bits in the *Block* field = 10**
Total number of address bits = 46
Therefore, **Number of bits in the *Tag* field =** 46 - 6 - 10 **= 30**

Problem No. 4

(a) Block size = 64 bytes = 2^6 bytes = 2^6 words
Cache size = 2M-byte = 2^{21} bytes
Number of cache blocks per set = 16 = 2^4
Number of sets = Cache size / (Block size * Number of blocks per set) = $2^{21} / (2^6 * 2^4) = 2^{11}$ = **2048**

(b) **Number of bits in the *Word* field = 6**
Number of bits in the *Set* field = 11
Total number of address bits = 46
Therefore, **Number of bits in the *Tag* field =** 46 - 6 - 11 **= 29**

Problem No. 5

Block size = 16 words = 2^4 words => No. of bits in the *Word* field = 4
Number of cache blocks = 8 = 2^3 => No. of bits in the *Block* field = 3
Total number of address bits = 10 => No. of bits in the *Tag* field = 10 – 4 - 3 = 3
For a given 10-bit address, the 3 most significant bits represent the *Tag*, the next 3 bits represent the *Block*, and the 4 least significant bits represent the *Word*.

(a) The cache is initially empty. Therefore, all the cache blocks are invalid.

Access # 1:
Address = $(20)_{10} = (0000010100)_2$
For this address, *Tag* = 000, *Block* = 001
Since the cache block 001 is empty before this access, this will be a cache **miss**
After this access, *Tag* field for cache block 001 is set to 000

Access # 2:
Address = $(32)_{10} = (0000100000)_2$
For this address, *Tag* = 000, *Block* = 010
Since the cache block 010 is empty before this access, this will be a cache **miss**
After this access, *Tag* field for cache block 010 is set to 000

Access # 3:
Address = $(512)_{10} = (1000000000)_2$
For this address, *Tag* = 100, *Block* = 000
Since the cache block 000 is empty before this access, this will be a cache **miss**
After this access, *Tag* field for cache block 000 is set to 100

Access # 4:
Address = $(160)_{10} = (0010100000)_2$
For this address, *Tag* = **001**, *Block* = 010

Current tag for cache block 010 = **000** (set in access # 2)
Address tag does not match the block tag => cache **miss**
After this access, *Tag* field for cache block 010 is set to 001

Access # 5:
Address = $(32)_{10} = (0000100000)_2$
For this address, *Tag* = **000**, *Block* = 010
Current tag for cache block 010 = **001** (set in access # 4)
Address tag does not match the block tag => cache **miss**
After this access, *Tag* field for cache block 010 is set to 000

Access # 6:
Address = $(896)_{10} = (1110000000)_2$
For this address, *Tag* = **111**, *Block* = 000
Current tag for cache block 000 = **100** (set in access # 3)
Address tag does not match the block tag => cache **miss**
After this access, *Tag* field for cache block 000 is set to 111

Access # 7:
Address = $(904)_{10} = (1110001000)_2$
For this address, *Tag* = **111**, *Block* = 000
Current tag for cache block 000 = **111** (set in access # 6)
Address tag matches the block tag => cache **hit**

Access # 8:
Address = $(512)_{10} = (1000000000)_2$
For this address, *Tag* = **100**, *Block* = 000
Current tag for cache block 000 = **111** (set in access # 6)
Address tag does not match the block tag => cache **miss**
After this access, *Tag* field for cache block 000 is set to 100

Cache hit ratio = Number of hits / Number of accesses = 1 / 8 = **0.125**

(b) The tag contents for the cache are shown below:

Block	000	001	010	011	100	101	110	111
Tag	100	000	000	Invalid	Invalid	Invalid	Invalid	Invalid

Problem No. 6

Block size = 16 words = 2^4 words => No. of bits in the *Word* field = 4
Number of cache blocks = 8 = 2^3
Cache blocks per set = 2 (2-way set-associative cache)
Therefore, number of sets = 8 / 2 = 4 = 2^2 => No. of bits in the *Set* field = 2
Total number of address bits = 10 => No. of bits in the *Tag* field = 10 – 4 – 2 = 4
For a given 10-bit address, the 4 most significant bits represent the *Tag*, the next 2 bits represent the *Set*, and the 4 least significant bits represent the *Word*.

The cache is initially empty. Therefore, all the cache blocks are invalid.

Access # 1:

Address = $(20)_{10} = (0000010100)_2$

For this address, *Tag* = 0000, *Set* = 01

Since the cache is empty before this access, this will be a cache **miss**

Cache contents after this access are:

Set	00		01		10		11	
Block	0	1	0	1	0	1	0	1
Tag	Invalid	Invalid	**0000**	Invalid	Invalid	Invalid	Invalid	Invalid

Access # 2:

Address = $(32)_{10} = (0000100000)_2$

For this address, *Tag* = 0000, *Set* = 10

Since the set "10" is empty before this access, this will be a cache **miss**

Cache contents after this access are:

Set	00		01		10		11	
Block	0	1	0	1	0	1	0	1
Tag	Invalid	Invalid	0000	Invalid	**0000**	Invalid	Invalid	Invalid

Access # 3:

Address = $(512)_{10} = (1000000000)_2$

For this address, *Tag* = **1000**, *Set* = 00

Since the set "10" is empty before this access, this will be a cache **miss**

Cache contents after this access are:

Set	00		01		10		11	
Block	0	1	0	1	0	1	0	1
Tag	**1000**	Invalid	0000	Invalid	0000	Invalid	Invalid	Invalid

Access # 4:

Address = $(160)_{10} = (0010100000)_2$

For this address, *Tag* = **0010**, *Set* = 10

Address tag does not match any of the block tags in set 10 => cache **miss**

New contents are inserted in block-1 of set 10. Cache contents after this access are:

Set	00		01		10		11	
Block	0	1	0	1	0	1	0	1
Tag	1000	Invalid	0000	Invalid	0000	0010	Invalid	Invalid

Access # 5:

Address = $(32)_{10} = (0000100000)_2$

For this address, *Tag* = **0000**, *Set* = 10

Current tag for cache block "0" in set 00 = **0000**

Address tag matches the block tag => cache **hit**. There is no change in cache contents.

Access # 6:

Address = $(896)_{10} = (1110000000)_2$

For this address, *Tag* = **1110**, *Set* = 00
Address tag does not match any of the block tags in set 00 => cache **miss**
New contents are inserted in block-1 of set 00. Cache contents after this access are:

Set	00		01		10		11	
Block	0	1	0	1	0	1	0	1
Tag	1000	1110	0000	Invalid	0000	0010	Invalid	Invalid

Access # 7:
Address = $(904)_{10} = (1110001000)_2$
For this address, *Tag* = **1110**, *Set* = 00
Current tag for cache block "1" in set 00 = **1110**
Address tag matches the block tag => cache **hit**. There is no change in cache contents.

Access # 8:
Address = $(512)_{10} = (1000000000)_2$
For this address, *Tag* = **1000**, *Set* = 00
Current tag for cache block "0" in set 00 = **1000**
Address tag matches the block tag => cache **hit**. There is no change in cache contents.

Cache hit ratio = Number of hits / Number of accesses = 3 / 8 = **0.375**