**The University of Alabama in Huntsville**
**ECE Department**
**CPE 431 01, CPE 531 01/01R**
**Fall 2022**
**Cache Memory Basics**

**Due October 18, 2022**
**1.0 (20), 2.0 (10), 3.0 (30), 4.0 (15)**

**1.0** **<5.1>** In this exercise we look at memory locality properties of matrix computation. The following code is written in MATLAB, where elements within the same column are stored contiguously. Assume each word is a 32-bit integer.

```
for I = 1:8
for J = 1:8000
    A(I, 1) = B(I, J) + A(J, I);
end
end
```

**1.0.1** For each variable, identify whether or not it exhibits temporal locality.
**1.0.2** For each variable, identify whether or not it exhibits spatial locality.

| Variable | Locality | Reason |
|---|---|---|
| A(I,1) | Temporal | Bc elements accessed are from the same column. Could also argue not since it changes every time in the inner loop |
| B(I,J) | Nothing | B(1,1), B(1,2), B(1,3)…..Bc different columns accessed each time and also different address each time. |
| A(J,I) | Spatial | Bc elements accessed from the same column |
| I | Temporal | Constantly needed |
| J | Temporal | Constantly needed |

**2.0 <5.3>** A cache has the following parameters: b, block size given in numbers of words; S, number of sets; N, number of ways; 2, byte offset; and A; number of address bits.

**2.0.1** In terms of the parameters, what is the cache capacity, C?

Block size in bytes * blocks per set * number of sets OR number of sets * number of ways * block size
N * S = total blocks (Set/block ration) * b(block/word ratio now) * byte Offset^2 ( word/bytes ratio)
Byte offset^2 bc byte offset = log2(bytes in word)

Cache Capacity = N*S*b*4

**2.0.2** What are S and N for a fully associate cache of capacity C words with block size b?

Fully associate means 1 set. S = 1
Cache capacity = Sets * ways * block size ->C = 1 * N * b -> N = C/b

**3.0 <5.3>** Caches are important to providing a high-performance memory hierarchy to processors. Below is a list of 32-bit hexadecimal memory addresses, given as byte addresses. 74, A0, 78, 38C, AC, 84, 88, 8C, 7C, 34, 38, 13C, 388, 18C

**3.0.1** For each of these references, identify the index and the tag, given a direct-mapped cache with 16 one-word blocks. List if each reference is a hit or a miss, assuming the cache is initially empty and show every entry to the cache, including the tag value and the addresses of all data items stored. Use hexadecimal or binary, whichever is easier.

Block = 1 word
Blocks = 16
Byte offset = 2
Block offset = none
Index = 4 found (log2(16))
32-12 = 26

Number of fill in 0's: 0000_0000_0000_0000_0000_

| Address(Hex) | Index (binary) | Tag | Hit/Miss | Set | Data |
|---|---|---|---|---|---|
| 74 | 1101 | 0000_0000_0000_0000_0000_0000_01 | Miss | 13 | M[74…77] |
| A0 | 1000 | 0000_0000_0000_0000_0000_0000_10 | Miss | 8 | M[A0…A3] |
| 78 | 1110 | 0000_0000_0000_0000_0000_0000_01 | Miss | 14 | M[78….7B] |
| 38C | 0011 | 0000_0000_0000_0000_0000_0011_10 | Miss | 3 | M[38C….38F] |
| AC | 1011 | 0000_0000_0000_0000_0000_0000_10 | Miss | 11 | M[AC…AF] |
| 84 | 0000 | 0000_0000_0000_0000_0000_0000_10 | Miss | 0 | M[84…87] |
| 88 | 0010 | 0000_0000_0000_0000_0000_0000_10 | Miss | 2 | M[88…8B] |
| 8C | 0011 | 0000_0000_0000_0000_0000_0000_10 | Miss | 3 | M[8C…8F] |
| 7C | 1111 | 0000_0000_0000_0000_0000_0000_01 | Miss | 15 | M[7C…7F] |
| 34 | 1101 | 0000_0000_0000_0000_0000_0000_00 | Miss | 11 | Update to M[34….37] |
| 38 | 1110 | 0000_0000_0000_0000_0000_0000_00 | Miss | 14 | Update to M[38…3B] |
| 13C | 1111 | 0000_0000_0000_0000_0000_0001_00 | Miss | 15 | Update to M[13C…13F] |
| 388 | 0010 | 0000_0000_0000_0000_0000_0011_10 | Miss | 2 | Update to M[388….38B] |
| 18C | 0011 | 0000_0000_0000_0000_0000_0001_10 | Miss | 3 | Update to M[18C…18F] |

**3.0.2** For each of these references, identify the index and the tag, given a direct-mapped cache with two word blocks and a total of 32 words. List if each reference is a hit or a miss, assuming the cache is initially empty and show every entry to the cache, including the tag value and the addresses of all data items stored. Use hexadecimal or binary, whichever is easier.

Block = 2 word
Number of Blocks = 32 words | 1 block/2 words = 16
Byte offset = 2
Block offset = 1
Index = 4
32 – (2+1+4) = 25 bits for a tag

| Address(Hex) | Index | Tag | Offsets | Set | Data | Hit/Miss |
|---|---|---|---|---|---|---|
| 74 | 1110 | 0000_0000_0000_0000_0000_0000_0 | 100 | 14 | M[70…77] | Miss |
| A0 | 0100 | 0000_0000_0000_0000_0000_0000_1 | 000 | 4 | M[A0…A7] | Miss |
| 78 | 1111 | 0000_0000_0000_0000_0000_0000_0 | 000 | 15 | M[78..7F] | Miss |
| 38C | 0001 | 0000_0000_0000_0000_0000_0011_1 | 100 | 1 | M[388..38F] | Miss |
| AC | 0101 | 0000_0000_0000_0000_0000_0000_1 | 100 | 5 | M[A8….AF] | Miss |
| 84 | 0000 | 0000_0000_0000_0000_0000_0000_1 | 100 | 0 | M[80…87] | Miss |
| 88 | 0001 | 0000_0000_0000_0000_0000_0000_1 | 000 | 1 | Update to M[88…8F] | Miss |
| 8C | 0001 | 0000_0000_0000_0000_0000_0000_1 | 100 | 1 | M[88…8F] | Hit |
| 7C | 1111 | 0000_0000_0000_0000_0000_0000_0 | 100 | 15 | M[78…7F] | Hit |
| 34 | 0110 | 0000_0000_0000_0000_0000_0000_0 | 100 | 6 | M[30…37] | Miss |
| 38 | 0111 | 0000_0000_0000_0000_0000_0000_0 | 000 | 7 | M[38…3F] | Miss |
| 13C | 0111 | 0000_0000_0000_0000_0000_0001_0 | 100 | 7 | Update to M[138…13F] | Miss |
| 388 | 0001 | 0000_0000_0000_0000_0000_0011_1 | 000 | 1 | Update to M[388…38F] | Miss |
| 18C | 0001 | 0000_0000_0000_0000_0000_0001_1 | 100 | 1 | Update to M[188…18F] | Miss |

**4.0** **<5.4>** You are building a computer with a hierarchical memory system that consists of separate instruction and data caches followed by main memory. You are using a MIPS single cycle processor running at 2.5 GHz.

**4.0.1** Suppose the instruction cache is perfect (i.e., always hits) but the data cache has a 7% miss rate. On a cache miss, the processor stalls for 45 ns to access main memory, then resumes normal operation. Taking cache misses into account, what is the average memory access time?

Avg memory access time = Hit time + miss rate * miss penalty
Hit time =1 cycle/2.5E9 cycles/s = 4E-10 s
AMAT = 4E-10+(0.07*45ns) = 3.55 ns

**4.0.2** 7

Memory Miss time = 3.55 ns or 8 cycles
Normal hit time = 0.4 ns or 1 cycle

Load takes 3.55 ns for memory access then another 4 cycles to complete (1.6ns)
Store is 3.55 ns + 3 cycles to complete (1.2)
Branch 3 cycles (1.22)
Other is 4 cycles (1.6)

*Unit Check*
Instructions * cycles where cycles are converted to seconds
Cycles/instructions = 0.25 instructions*((4 cycles*1 second/2.5E9 cycles )+ 3.55 ns)

*Solution*
Average CPI = 0.25(1.6+3.55) + 0.15(1.2+3.55) + 0.10(1.2) + 0.52(1.6)
= 1.2875 + 0.7125 + 0.12 + 0.832
= 2.952 ns which is 7.38 cycles because 2.875ns|1cycle/0.4ns = cycles

**4.0.3** Now suppose the instruction cache is also non-ideal and has a 3% miss rate. What is the average CPI for the benchmark in 4.02? Take into account both instruction and data cache misses.

AMAT = hit time + miss rate * miss penalty for both data and instruction
Miss Rate = 0.07
Miss penalty = 45 ns
Hit time = 2.952 ns

CPI for instruction cache = 2.952 ns + (0.07*45ns) = 4.302 ns or about 10.755 cycles

Average CPI for data and instruction cache = 4.302 + 3.55 = 7.852 ns or about 19.63 cycles