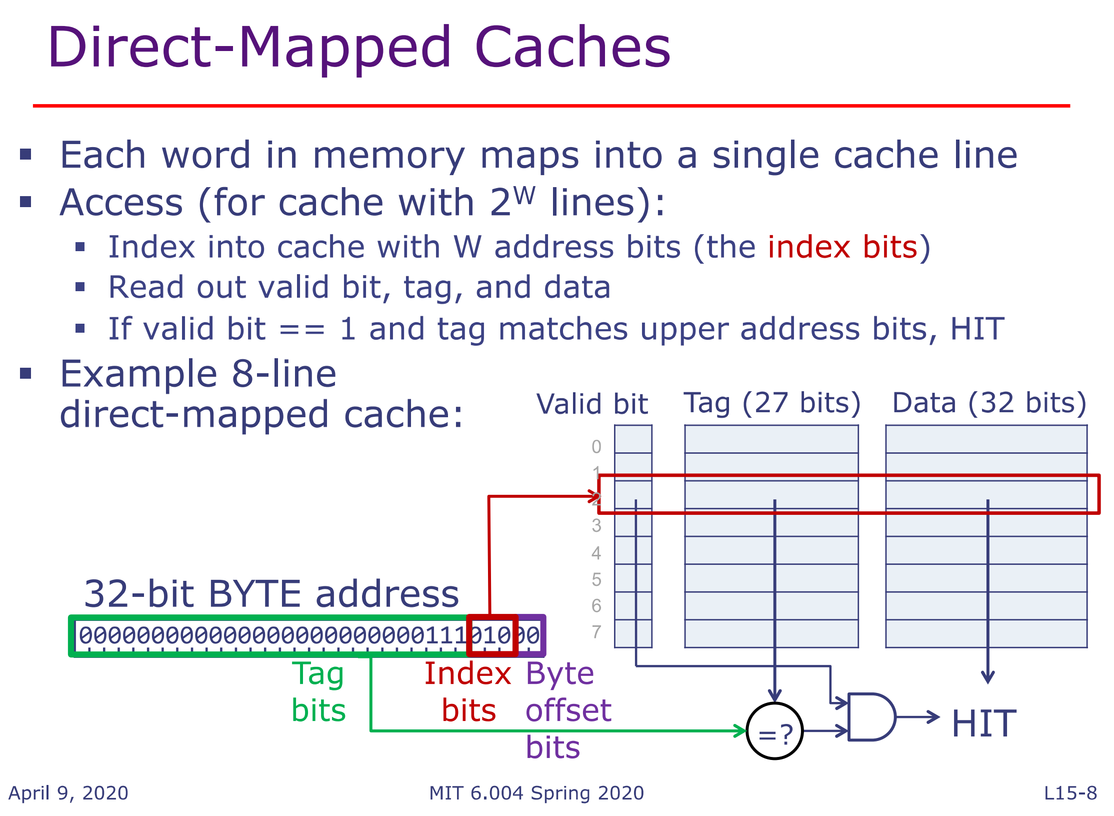
6.004 Recitation Problems
L15 – Caches

Keep the most often-used data in a small, fast SRAM (often local to CPU chip). The reason this strategy works: LOCALITY.

– *Temporal locality: If a location has been accessed recently, it is likely to be accessed (reused) soon*
– *Spatial locality: If a location has been accessed recently, it is likely that nearby locations will be accessed soon*

AMAT(Average Memory Access Time) = HitTime + MissRatio * MissPenalty



Direct-Mapped Caches

- Each word in memory maps into a single cache line
- Access (for cache with 2^W lines):
  - Index into cache with W address bits (the index bits)
  - Read out valid bit, tag, and data
  - If valid bit == 1 and tag matches upper address bits, HIT
- Example 8-line direct-mapped cache:

Valid bit
Tag (27 bits)
Data (32 bits)
0
1
2
3
4
5
6
7
32-bit BYTE address
00000000000000000000000011101000
Tag bits
Index bits
Byte offset bits
=?
HIT

April 9, 2020 MIT 6.004 Spring 2020 L15-8

Example: Direct-Mapped Caches

64-line direct-mapped cache → 64 indices → 6 index bits

Read Mem[0x400C]

$\underbrace{0100\ 0000}\ \underbrace{0000\ 11}\underbrace{00}$

TAG: 0x40
INDEX: 0x3
BYTE OFFSET: 0x0

HIT, DATA 0x42424242

Would 0x4008 hit?

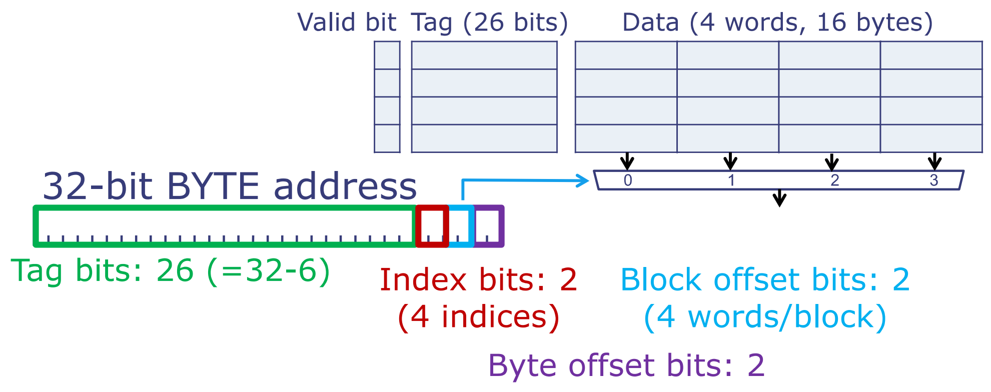
INDEX: 0x2 ▯ tag mismatch
▯ MISS

	Valid bit	Tag (24 bits)	Data (32 bits)
0	1▯	0x000058▯	0xDEADBEEF▯
1	1▯	0x000058▯	0x00000000▯
2	1▯	0x000058▯	0x00000007▯
3	1▯	0x000040▯	0x42424242▯
4	0▯	0x000007▯	0x6FBA2381▯
	⋮	⋮	⋮
63	1▯	0x000058▯	0xF7324A32▯

Part of the address (index bits) is encoded in the location
Tag + Index bits unambiguously identify the data's address

Block Size

- Take advantage of spatial locality: Store multiple words per data line
  - Always fetch entire block (multiple words) from memory
  - Another advantage: Reduces size of tag memory!
  - Potential disadvantage: Fewer indices in the cache
- Example: 4-block, 16-word direct-mapped cache



Problem 1. ★

(A) The timing for a particular cache is as follows: checking the cache takes 1 cycle. If there's a hit the data is returned to the CPU at the end of the first cycle. If there's a miss, it takes 10 *additional* cycles to retrieve the word from main memory, store it in the cache, and return it to the CPU. If we want an average memory access time of 1.4 cycles, what is the minimum possible value for the cache's hit ratio?

Minimum possible value of hit ratio: ____

(B) If the cache block size, i.e., words/cache line, is doubled but the total number of data words in the cache is unchanged, how will the following cache parameters change? Please circle the best answer.

# of block offset bits: UNCHANGED … +1 … -1 … 2x … 0.5x … CAN'T TELL

# of tag bits: UNCHANGED … +1 … -1 … 2x … 0.5x … CAN'T TELL

# of cache lines: UNCHANGED … +1 … -1 … 2x … 0.5x … CAN'T TELL

Consider a direct-mapped cache with 64 total data words with 1 word/cache line. This cache architecture is used for parts (C) through (F).

(C) If cache line number 5 is valid and its tag field has the value 0x1234, what is the address in main memory of the data word currently residing in cache line 5?

Main memory address of data word in cache line 5: _______

The program shown on the right repeatedly executes an inner loop that sums the 16 elements of an array that is stored starting in location 0x310.

The program is executed for many iterations, then a measurement of the cache statistics is made during one iteration through all the code, i.e., starting with the execution of the instruction labeled `outer_loop:` until just before the next time that instruction is executed.

```
  . = 0            // tell assembler to start at
                   // address 0
outer_loop:
  addi x4, x0, 16  // initialize loop index J
  mv x1, x0        // x1 holds sum, initially 0

loop:              // add up elements in array
  subi x4, x4, 1   // decrement index
  slli x2, x4, 2   // convert to byte offset
  lw x3, 0x310(x2) // load value from A[J]
  add x1, x1, x3   // add to sum
  bne x4, x0, loop // loop until all words are summed

  j outer_loop     // perform test again!
```

(D) In total, how many instruction fetches occur during one complete iteration of the outer loop? How many data reads?

Number of instruction fetches: _______

Number of data reads: _______

(E) How many instruction fetch misses occur during one complete iteration of the outer loop? How many data read misses? Hint: remember that the array starts at address 0x310.

Number of instruction fetch misses: ________

Number of data read misses: ________

(F) What is the hit ratio measured after one complete iteration of the outer loop?

Hit ratio: ___________________

Problem 2.

The RISC-V Engineering Team is working on the design of a cache. They've decided that the cache will have **a total of 2^{10} = 1024 data words**, but are still thinking about the other aspects of the cache architecture.

First assume the team chooses to build a direct-mapped cache with a block size of 4 words.

(A) Please answer the following questions:

Number of lines in the cache: _____

Number of bits in the tag field for each cache entry: _____

(B) This cache takes *2 clock cycles* to determine if a memory access is a hit or a miss and, if it's a hit, return data to the processor. If the access is a miss, the cache takes *20 additional clock cycles* to fill the cache line and return the requested word to the processor. If the hit rate is 90%, what is the processor's average memory access time in clock cycles?

Average memory access time assuming 90% hit rate (clock cycles): _____

Now assume the team chooses to build a 2-way set-associative write-back cache with a block size of 4 words. *The total number of data words in the entire cache is still 1024*. The cache uses a LRU replacement strategy.

(C) Please answer the following questions

Address bits used as block offset: A[_:_]

Address bits used as cache line index: A[_:_]

Address bits used for tag comparison: A[_:_]

(D) To implement the LRU replacement strategy this cache requires some additional state for each set. How many state bits are required for each set?

Number of state bits needed for each set for LRU: _____

To test this set-associative cache, the team runs the benchmark code shown on the right. The code sums the elements of a 16-element array. The first instruction of the code is at location 0x0 and the first element of the array is at location 0x10000. Assume that the cache is empty when execution starts and remember *the cache has a block size of 4 words*.

```
. = 0x0
   mv x3, x0 // index
   mv x1, x0 // sum
   // x4 = 0x10000
   lui x4, 0x10

L: add x5, x4, x3
   lw x2, 0(x5)
   add x1, x1, x2
   addi x3, x3, 4
   slti x2, x3, 64
   bnez x2, L
   unimp    // halt

. = 0x10000
A: .word 0x1
   .word 0x2
   …
   .word 0xF
   .word 0x10
```

(E) How many instruction misses will occur when running the benchmark?

Number of instruction misses when running the benchmark: _____

(F) How many data misses (i.e., misses caused by the memory access from the LD instruction) will occur when running the benchmark?

Number of data misses when running the benchmark: _____

(G) What's the exact hit rate when the complete benchmark is executed?

Benchmark hit rate: _____

Problem 3.

Assume, the program shown on the right is being run on a RISC-V processor with a cache with the following parameters:

- **2-way set-associative**
- **block size of 2**, i.e., 2 data words are stored in each cache line
- total number of data words in the cache is **32**
- **LRU** replacement strategy

```
. = 0x240          // start of program
test:
  addi x4, x0, 16 // initialize loop index J
                  // to size of array
  mv x1, x0       // x1: sum

loop:             // add up elements in array
  subi x4, x4, 1  // decrement index
  slli x2, x4, 2  // convert to byte offset
  lw x3, 0x420(x2)// load value from A[J]
  add x1, x1, x3  // add to sum
  bnez x4, loop   // loop N times

  j test          // perform test again!

// allocate space to hold array
. = 0x420
A: .word A[0]
   .word A[1]
   …
```

(A) The cache will divide the 32-bit address supplied by the processor into four fields: 2 bits of byte offset, B bits of block offset, L bits of cache line index, and T bits of tag field. Based on the cache parameters given above, what are the appropriate values for B, L, and T?

value for B: _____
value for L: _____
value for T: _____

(B) If the SLLI instruction is resident in a cache line, what will be its cache line index? the value of the tag field for the cache?

Cache line index for SLLI when resident in cache: _____

Tag field for SLLI when resident in cache: _____

(C) Given that the code begins at address 0x240 and the array begins at address 0x420, and that there are 16 elements in the array as shown in the code above, list *all* the values j (0 ≤ j < 16) where the location holding the value A[j] will map to the same cache line index as the `SLLI` instruction in the program.

List all j where A[j] have the same cache line index as SLLI: _____

(D) If the outer loop is run many times, give the steady-state hit ratio for the cache, i.e., assume that the number of compulsory misses as the cache is first filled are insignificant compared to the number of hits and misses during execution.

Steady-state hit ratio (%): _____

Problem 4. ★

Consider a 2-way set-associative cache where each way has 4 cache lines with a **block size of 2 words**. Each cache line includes a valid bit (V) and a dirty bit (D), which is used to implement a write-back strategy. The replacement policy is least-recently-used (LRU). The cache is used for both instruction fetch and data (LD,ST) accesses. Please use this cache when answering questions (A) through (D).

(A) Using this cache, a particular benchmark program experiences an average memory access time (AMAT) of 1.3 cycles. The access time on a cache hit is 1 cycle; the miss penalty (i.e., additional access time) is 10 cycles. What is the hit ratio when running the benchmark program? You can express your answer as a formula if you wish:

Hit ratio for benchmark program: _____

(B) The circuitry for this cache uses various address bits as the block offset, cache line index and tag field. Please indicate which address bits A[31:0] are used for each purpose by placing a "B" in each address bit used for the block offset, "L" in each address bit used for the cache line index, and "T" in each address bit used for the tag field.

Fill in each box with "B", "L", or "T"

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
																														0	0

(C) This cache needs room to store new data and based on the LRU replacement policy has chosen the cache line whose information is shown to the right for replacement. Since the current contents of that line are marked as dirty (D = 1), the cache must write some information back to main memory. What is the address of each memory location to be written? Please give each address in hex.

Way: 0
Cache line index: 3
Valid bit (V): 1
Dirty bit (D): 1
Tag field: 0x123

Addresses of each location to be written (in hex): _____

(D) This cache is used to run the following benchmark program. The code starts at memory address 0; the array referenced by the code has its first element at memory address 0x200. First determine the number of memory accesses (both instruction and data) made during each iteration through the loop. Then estimate the steady-state average hit ratio for the program, i.e., the average hit ratio after many iterations through the loop.

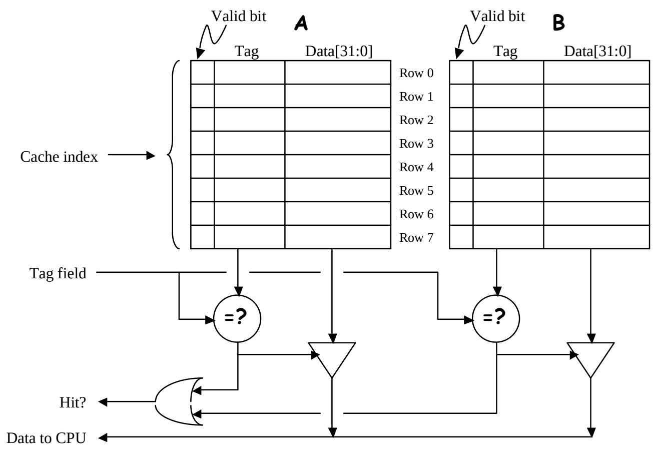
```
. = 0
    mv x3, x0           // byte index into array
    mv x1, x0           // initialize checksum accumulator
loop:
    lw x2, 0x200(x3)    // load next element of array
    slli x1, x1, 1       // shift checksum
    addi x1, x1, 1       // increment checksum
    add x1, x1, x2       // include data value in checksum
    addi x3, x3, 4       // byte index of next array element
    slti x2, x3, 1000    // process 250 entries
    bnez x2, loop
    unimp                // halt

. = 0x200
array:
    … array contents here …
```

Number of memory accesses made during each iteration of the loop: _____

Estimated steady-state average hit ratio: _____

Problem 5.



Consider the diagram to the right for a 2-way set associative cache to be used with our RISC-V processor. Each cache line holds a single 32-bit word of data along with its associated tag and valid bit (0 when the cache line is invalid, 1 when the cache line is valid).

(A) The RISC-V produces 32-bit byte addresses, A[31:0]. To ensure the best cache performance, which address bits should be used for the cache index? For the tag field?

address bits used for cache index: A[_:_]

address bits used for tag field: A[_:_]

(B) Suppose the processor does a read of location 0x5678. Identify which cache location(s) would be checked to see if that location is in the cache. For each location specify the cache section (A or B) and row number (0 through 7). E.g., **3A** for row 3, section A. If there is a cache hit on this access what would be the contents of the tag data for the cache line that holds the data for this location?

cache location(s) checked on access to 0x5678: _____

cache tag data on hit for location 0x5678 (hex): _____

(C) Assume that checking the cache on each read takes 1 cycle and that refilling the cache on a miss takes an *additional* 8 cycles. If we wanted the *average* access time over many reads to be 1.1 cycles, what is the minimum hit ratio the cache must achieve during that period of time? You needn't simplify your answer.

minimum hit ratio for 1.1 cycle average access time: _____

(D) Estimate the approximate cache hit ratio for the following program. Assume the cache is empty before execution begins (all the valid bits are 0) and that an LRU replacement strategy is used. Remember the cache is used for both instruction and data (LD) accesses.

```
      . = 0
      addi x4, x0, 0x100
      mv x1, x0
      lui x2, 1          // x2 = 0x1000
loop: lw x3, 0(x4)
      addi x4, x4, 4
      add x1, x1, x3
      addi x2, x2, -1
      bnez x2, loop
      sw x1, 0x100(x0)
      unimp              // halt

      . = 0x100
```

```
source:
      . = . + 0x4000 // Set source to 0x100, reserve 0x1000 words
```

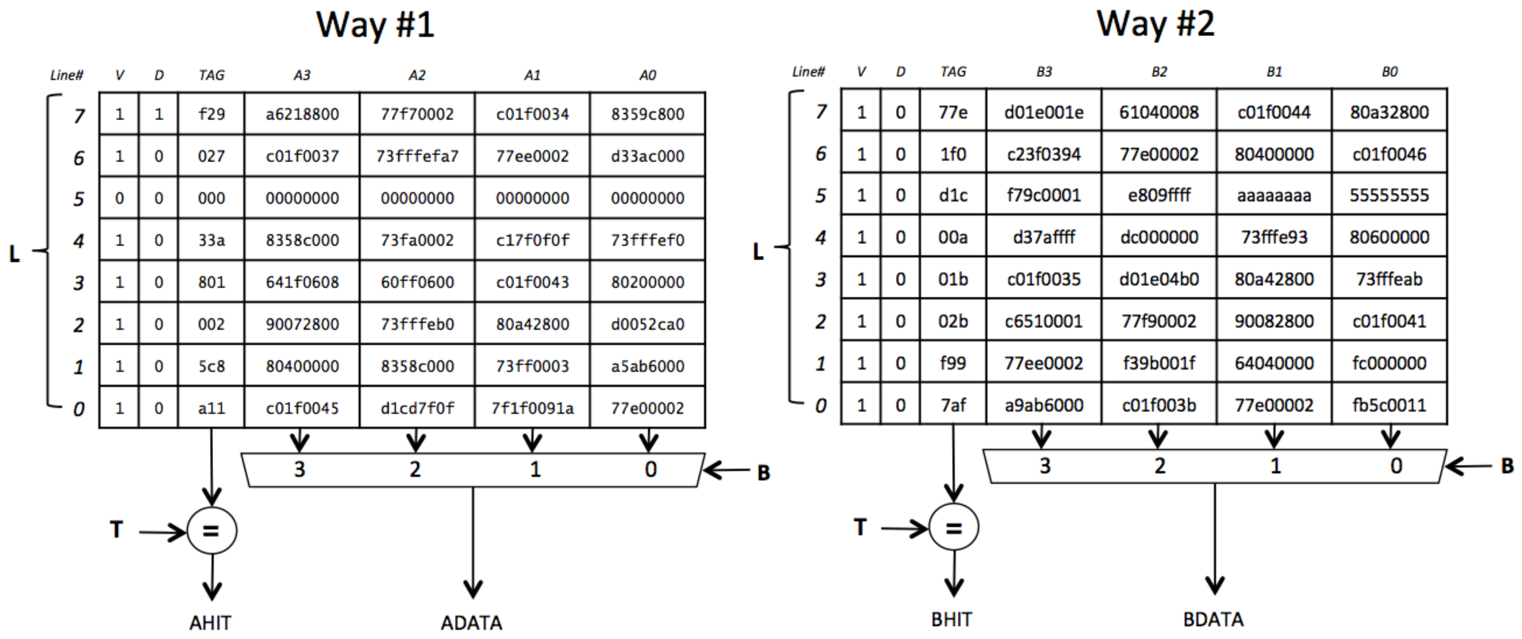
approximate hit ratio: _____

(E) After the program of part (D) has finished execution what information is stored in row 4 of the cache? Give the addresses for the two locations that are cached (one in each of the sections) or briefly explain why that information can't be determined.

Addresses whose data is cached in "Row 4": _____ and _____

Problem 6. ★

A standard unpipelined RISC-V is connected to a 2-way set-associative cache containing 8 sets, with a block size of 4 32-bit words. The cache uses a LRU replacement strategy. At a particular point during execution, a snapshot is taken of the cache contents, which are shown below. All values are in hex; **assume that any hex digits not shown are 0**.



Way #1

Line#	V	D	TAG	A3	A2	A1	A0
7	1	1	f29	a6218800	77f70002	c01f0034	8359c800
6	1	0	027	c01f0037	73fffefa7	77ee0002	d33ac000
5	0	0	000	00000000	00000000	00000000	00000000
4	1	0	33a	8358c000	73fa0002	c17f0f0f	73fffef0
3	1	0	801	641f0608	60ff0600	c01f0043	80200000
2	1	0	002	90072800	73fffeb0	80a42800	d0052ca0
1	1	0	5c8	80400000	8358c000	73ff0003	a5ab6000
0	1	0	a11	c01f0045	d1cd7f0f	7f1f0091a	77e00002

Way #2

Line#	V	D	TAG	B3	B2	B1	B0
7	1	0	77e	d01e001e	61040008	c01f0044	80a32800
6	1	0	1f0	c23f0394	77e00002	80400000	c01f0046
5	1	0	d1c	f79c0001	e809ffff	aaaaaaaa	55555555
4	1	0	00a	d37affff	dc000000	73fffe93	80600000
3	1	0	01b	c01f0035	d01e04b0	80a42800	73fffeab
2	1	0	02b	c6510001	77f90002	90082800	c01f0041
1	1	0	f99	77ee0002	f39b001f	64040000	fc000000
0	1	0	7af	a9ab6000	c01f003b	77e00002	fb5c0011

(A) The cache uses bits from the 32-bit byte address produced by the processor to select the appropriate set (L), as input to the tag comparisons (T) and to select the appropriate word from the data block (B). For correct and optimal performance what are the appropriate portions of the address to use for L, T and B? Express your answer in the form "A[N:M]" for N and M in the range 0 to 31, or write "CAN'T TELL".

Address bits to use for L: A[_:_]
Address bits to use for T: A[_:_]
Address bits to use for B: A[_:_]

(B) For the following addresses, if the contents of the specified location appear in the cache, give the location's 32-bit contents in hex (determined by using the appropriate value from the cache). If the contents of the specified location are NOT in the cache, write "MISS".

Contents of location 0xA1100 (in hex) or "MISS": _____
Contents of location 0x548 (in hex) or "MISS": _____

(C) Ignoring the current contents of the cache, is it possible for the contents of locations 0x0 and 0x1000 to both be present in the cache simultaneously?

Locations 0x0 and 0x1000 present simultaneously (circle one): YES … NO

(D) Give a one-sentence explanation of how the D bit got set to 1 for Line #7 of Way #1. At what point should the D bit be reset to 0?

One sentence explanation

(E) The following code snippet sums the elements of the 32-element integer array X. Assume this code is executing on a RISC-V processor with a cache architecture as described above and that, initially, the cache is empty, i.e., all the V bits have been set to 0. Compute the hit ratio as this program runs until it executes the *unimp* instruction, a total of 2 + (6*32) + 1 = 195 instruction fetches and 32 data accesses.

Hit ratio: _____

```
    . = 0
    mv x4, x0           // loop counter
    mv x1, x0           // accumulated sum

loop:
    slli x2, x4, 2      // convert loop counter to byte offset
    lw x3, 0x100(x2)    // load next value from array
    add x1, x1, x3      // add value to sum
    addi x4, x4, 1      // increment loop counter
    slti x2, x4, 32     // finished with all 32 elements?
    bnez x2, loop       // nope, keep going

    unimp               // all done, sum in x1

. = 0x100
X:  .word 1             // the 32-element integer array X
    .word 2
    …
    .word 32
```

Problem 7. ★

After his geek hit single *I Hit the Line*, renegade singer Johnny Cache has decided he'd better actually learn how a cache works. He bought three RISC-V processors, identical except for their cache architectures:

- ***Proc1*** has a 64-line direct-mapped cache
- ***Proc2*** has a 2-way set associative cache, LRU, with a total of 64 lines
- ***Proc3*** has a 4-way set associative cache, LRU, with a total of 64 lines

Note that each cache has the same total capacity: 64 lines, each holding a single 32-bit **word** of data or instruction. All three machines use the same cache for data and instructions fetched from main memory.

Johnny has written a simple test progr

```
// Try a little cache benchmark
// Assume x7 = 0x2000 (data region A)
// Assume x8 = 0x3000 (data region B)
// Assume x9 = 16 (size of data regions in BYTES!)

. = 0x1000                    // start program here
P:    addi x6, x0, 1000       // outer loop count
Q:    mv x3, x9               // Loop index i (array offset)
R:    addi x3, x3, -4         // i = i-1
      addi x9, x3, x7         // x9 = address of A[i]
      addi x10, x3, x8        // x10 = address of B[i]
      lw x1, 0(x9)            // read A[i]
      lw x2, 0(x10)           // read B[i]
      bnez x3, R
      addi x6, x6, -1         // repeat many times
      bnez x6, Q
      unimp                   // halt
```

Johnny runs his program on each processor, and finds that one processor model outperforms the other two.

(A) Which processor model gets the highest hit ratio on the above benchmark?

Circle one: Proc1 Proc2 Proc3

(B) Johnny changes the value of **B** in his program to **0x2000** (same as **A**), and finds a substantial improvement in the hit rate attained by one of the processor models (approaching 100%). Which model shows this marked improvement?

Circle one: Proc1 Proc2 Proc3

(C) Finally, Johnny moves the code region to 0x0 and the two data regions **A**, and **B** each to **0x0**, and sets **x9** to **64**. What is the TOTAL number of cache misses that will occur executing this version of the program on each of the processor models?

TOTAL cache misses running on Proc1: _____ ; Proc2: _____ ; Proc3: _____