# CSE 120 Spring 2021
# Homework Assignment 1

Jaden Liu
zliu259@ucsc.edu
University of California at Santa Cruz
Santa Cruz, CA 95064 USA

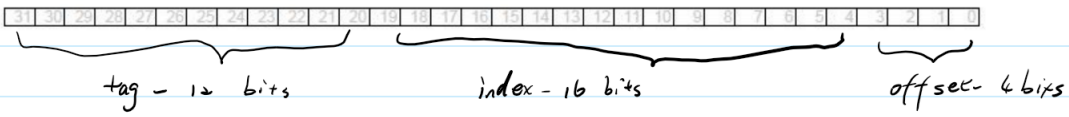
June 6, 2021

## 1 HW1

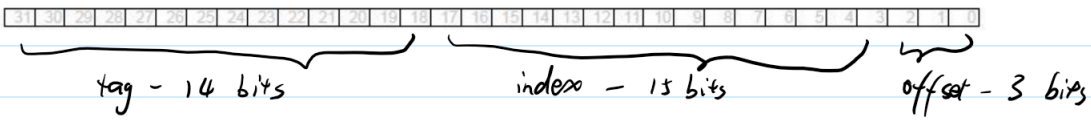
### Question 1

For each of the cache configurations listed below, show the decomposition of a 32 bit address to be used with it (tag/index/offset).

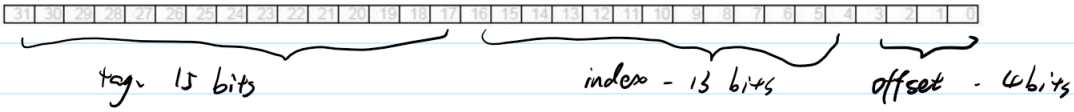
a.




b.




c.




### Question 2

(a)Fill in the table below and derive the hit rate for the given memory reference stream for a 512KB (Kilo Byte) 2-way associative cache with an 16-byte cache line? Assume MRU if needed. **(2 point)**

*Solution for a.*

| Memory Address | Tag | Index | Byte | Hit/Miss |
|---|---|---|---|---|
| 0x1234AB22 | 0x48D | 0xAB2 | 0x2 | Miss |
| 0x1234AB20 | 0x48D | 0xAB2 | 0x0 | Hit |
| 0x1234D020 | 0x48D | 0xD02 | 0x0 | Miss |
| 0x1234D02F | 0x48D | 0xD02 | 0xF | Hit |
| 0x1234AB26 | 0x48D | 0xAB2 | 0x6 | Hit |
| 0x1234AB33 | 0x48D | 0xAB3 | 0x3 | Miss |
| 0x1234D023 | 0x48D | 0xD02 | 0x3 | Hit |
| 0x1234D02B | 0x48D | 0xD02 | 0xB | Hit |
| 0x1234AB22 | 0x48D | 0xAB2 | 0x2 | Hit |
| 0x1234AB28 | 0x48D | 0xAB2 | 0x8 | Hit |

Hit rate = 7/10 □

(b) If we opt for a fully associative cache instead (other parameters are the same), will we improve the hit rate? Fill the table as you did for the previous question, and derive the hit rate. Assume LRU if needed **(1 point)**

*Solution for b.*

| Memory Address | Tag | Index | Byte | Hit/Miss |
|---|---|---|---|---|
| 0x1234AB22 | 0x1234AB2 | - | 0x2 | Miss |
| 0x1234AB20 | 0x1234AB2 | - | 0x0 | Hit |
| 0x1234D020 | 0x1234D02 | - | 0x0 | Miss |
| 0x1234D02F | 0x1234D02 | - | 0xF | Hit |
| 0x1234AB26 | 0x1234AB2 | - | 0x6 | Hit |
| 0x1234AB33 | 0x1234AB3 | - | 0x3 | Miss |
| 0x1234D023 | 0x1234D02 | - | 0x3 | Hit |
| 0x1234D02B | 0x1234D02 | - | 0xB | Hit |
| 0x1234AB22 | 0x1234AB2 | - | 0x2 | Hit |
| 0x1234AB28 | 0x1234AB2 | - | 0x8 | Hit |

Hit rate = 7/10 □

## Question 3

We have a RISCV CPU with SV39 virtually indexed and virtually tagged caches. Fill in the table below for a 256 KB 2-way cache. Write the state of the tag bank (tag,valid bit, dirty bit) and if triggered a line fill and/or a line displacement. Notice that this question is not asking you to show the address breakdown like in the previous questions, but instead, show the contents in some fields of the cache **after** the memory operation is performed. Assume LRU if needed.

*Solution.*

| Memory Address | Tag | Valid bit | Dirty bit | Line Displacement | Hit/Miss | Relevant index |
|---|---|---|---|---|---|---|
| LD 0x1234AB22 | 0x91A | 1 | 0 | 1 | Miss | 0xAB2 |
| ST 0x1234AB20 | 0x91A | 1 | 1 | 0 | Hit | 0xAB2 |
| LD 0x1234D020 | 0x91A | 1 | 0 | 1 | Miss | 0xD02 |
| ST 0x1234D02F | 0x91A | 1 | 1 | 0 | Hit | 0xD02 |
| LD 0x1234AB26 | 0x91A | 1 | 1 | 0 | Hit | 0xAB2 |
| ST 0x1234AB33 | 0x91A | 1 | 1 | 1 | Miss | 0xAB3 |
| LD 0x1234D023 | 0x91A | 1 | 1 | 0 | Hit | 0xD02 |
| ST 0x1234D02B | 0x91A | 1 | 1 | 0 | Hit | 0xD02 |
| LD 0x1234AB22 | 0x91A | 1 | 1 | 0 | Hit | 0xAB2 |
| ST 0x1234AB28 | 0x91A | 1 | 1 | 0 | Hit | 0xAB2 |

□

## Question 4

Assume we have a CPU where the CPI is 1.35 if all the memory accesses hit in the cache. We run a memory intensive benchmark where the only data accesses are loads and stores, accounting for 50% of the total instructions. The miss penalty in this system is 25 clock cycles and the miss-rate is 3.5%.

a. How much faster is the CPU if all the instructions are cache hits? **(1 points)**

*Solution for a.* $CPI_{old} = 1.35$
$CPI_{new} = 1.35 + 0.5 * 0.035 * 25 = 1.7875$
$speed\_up = CPI_{new}/CPI_{old} = 1.324$

□

b. What is the speedup (Execute Time part a vs part b) if we increase the miss rate to 4% but reduce the miss penalty to 12 cycles? **(1 points)**

*Solution for b.* $CPI_{old} = 1.7875$
$CPI_{new} = 1.35 + 0.5 * 0.04 * 12 = 1.59$
$speed\_up = CPI_{new}/CPI_{old} = 0.890$

□

## Question 5

What is the AMAT for a system with the following specifications?

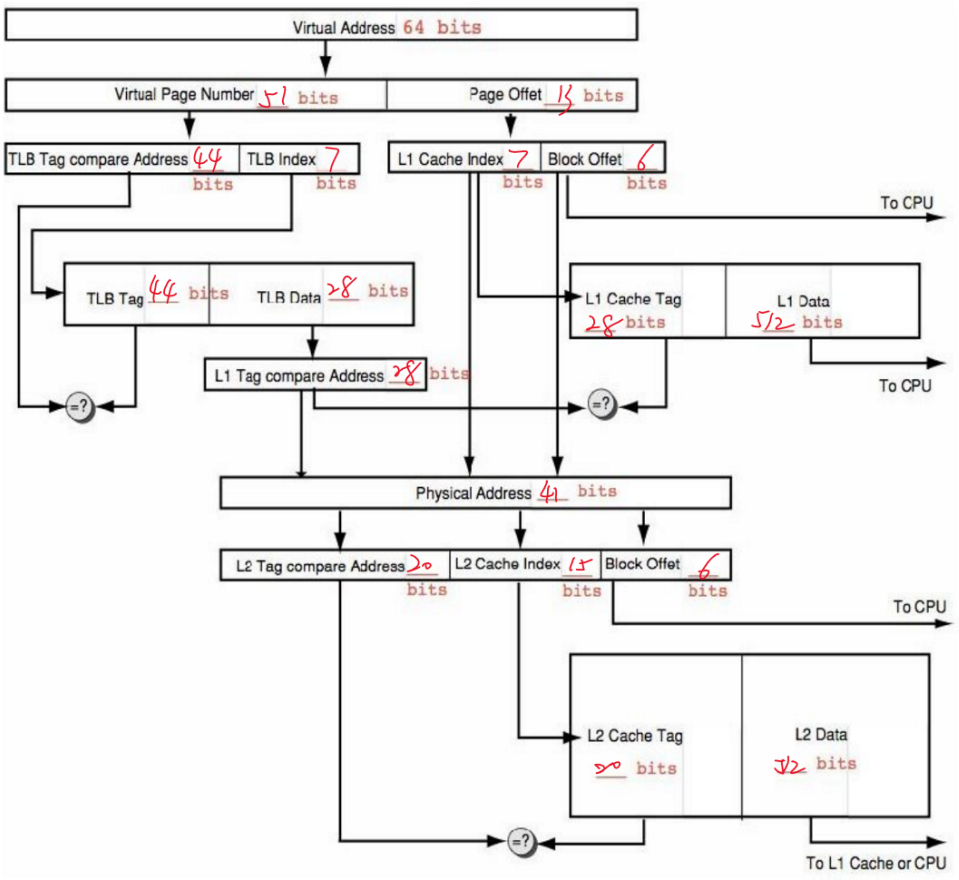
- Percentage of instructions accessing memory = 36%
- Main Memory Access Time = 30 ns
- Size of the L1 cache = 4 KB
- L1 cache hit rate = 60%
- L1 hit time = 0.6ns
- Size of the L2 cache = 10 MB
- L2 cache hit rate = 20%
- L2 hit time = 18ns
- core frequency 3.333GHz

Show your calculations for full credit. **(2 points)**

*Solution.* $AMAT = 0.6 + 0.4 * 18 + 0.4 * 0.8 * 30 = 17.4ns$ □

## Question 6

For the following figure shows a hypothetical memory hierarchy going from a virtual address to L2 cache access. The page size is 8KB, the TLB is direct mapped with 128 entries. The L1 cache is a direct mapped 8 KB, and the L2 cache is 2MB and direct mapped. Both use 64 byte blocks. The virtual address is 64 bits and the physical address is 41 bits. For each block in the figure below, fill in the number of bits that are used for all the translations, comparisons , indices, etc. in all the places indicated.


Virtual Address 64 bits
Virtual Page Number 51 bits
Page Offet 13 bits
TLB Tag compare Address 44 bits
TLB Index 7 bits
L1 Cache Index 7 bits
Block Offet 6 bits
To CPU
TLB Tag 44 bits
TLB Data 28 bits
L1 Cache Tag 28 bits
L1 Data 512 bits
To CPU
L1 Tag compare Address 28 bits
=?
=?
Physical Address 41 bits
L2 Tag compare Address 20 bits
L2 Cache Index 15 bits
Block Offet 6 bits
To CPU
L2 Cache Tag 20 bits
L2 Data 512 bits
=?
To L1 Cache or CPU

## Question 7

Given a 4 core processor with the following memory system configuration:

- 16-bit address space
- 16-bit word
- Private L1 per core (4 L1 caches total)
- Processor is cache coherent using MESI coherence protocol
- L1 size: 256 bytes
- L1 is direct mapped
- Shared L2 (shared between all 4 cores)
- L2 size: 1024 bytes
- L2 is 2-way set associative
- The replacement policy is LRU
- line size (for L1 and L2): 16 bytes
- L1 and L2 are inclusive
- For simplicity assume processor works with physical addresses only (so you can ignore TLB and virtual to physical page translation).
- For simplicity assume all accesses are to a single byte so that you can ignore alignment issues

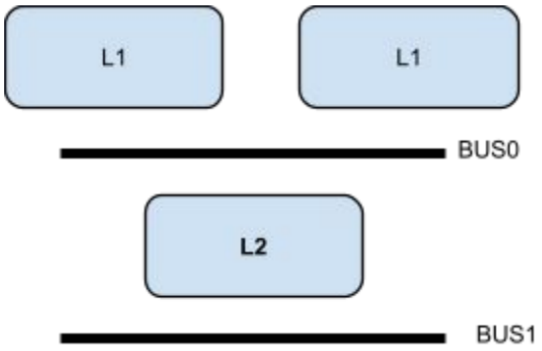
Complete the following table (next page) with information about the sequence of memory accesses. Note that values are listed in hex. Also the access are numbered to help make it more convenient for referring to them when showing your work. There are 4 cores in the processor labeled, A, B, C, and D. So each access lists which core it is for and which byte the processor is trying to access. **(2 point)**

*Solution.*

| Access | Core | Type | Address | L1 | L1 State | Bus | L2 | L2 |
|---|---|---|---|---|---|---|---|---|
| 1 | A | LD | 0xFFA0 | Miss | I->E | BR | Miss | I,I->E,I |
| 2 | A | ST | 0xFFA0 | Hit | E->M | – | – | – |
| 3 | B | ST | 0xFFA2 | Miss | I->M | BX,WB | Hit | E,I->M,I |
| 4 | A | LD | 0xFFA4 | Miss | I->S | BR,WB | Hit | M,I |
| 5 | C | LD | 0xFFAA | Miss | I->S | BR | Hit | M,I |
| 6 | C | ST | 0x01AA | Miss | S->M | BX | Miss | M,I->M,E |
| 7 | D | LD | 0x01AA | Miss | I->S | BR,WB | Hit | M,E->M,M |
| 8 | D | LD | 0x03AA | Miss | S->E | BR | Miss | M,M->E,M |
| 9 | A | LD | 0xFFA0 | Miss | I->E | BR | Miss | E,M->E,E |
| 10 | A | ST | 0x01AA | Miss | I->M | BX | Miss | E,E |
| 11 | A | LD | 0xFFA2 | Miss | M->E | BR | Hit | E,E->M,E |
| 12 | C | LD | 0x01A4 | Miss | I->E | BR | Hit | M,E |

## Question 8

There are two cores that perform **LD** and **ST**. Each core has a 512 byte direct mapped caches with 16 byte cache lines. All the lines are marked invalid before starting execution. In this question, there is a shared L2 cache and a "BUS" between the L1 caches and the L2 cache. The L2 cache has 2048 direct mapped 16 bytes cache line.




Update the table indicating the bus messages generated (BR, BX, WB) as a result of each operation, and the resulting cache line state. For the bus message indicate the processor source. (E.g: WB0 to indicate that processor 0's L1 cache did a write back). Follow the MESI protocol. **(2 point)**

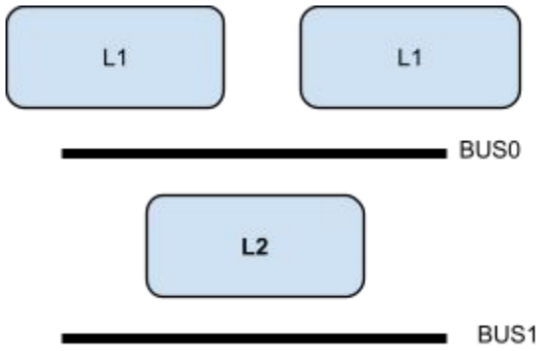
*Solution.*

| Operation | Bus0 | Bus1 | P0 L1 | P1 L1 | L2 |
|---|---|---|---|---|---|
| P0: ld [0x430] | BR0 | BR | E | I | E |
| P0: ld [0x438] | - | - | E | I | E |
| P1: ld [0x300] | BR1 | BR | I | E | E |
| P0: st [0x300] | BX0 | - | M | I | E |
| P1: ld [0x300] | BR1, WB0 | - | S | S | M |
| P0: ld [0x430] | - | - | E | I | E |
| P1: ld [0x400] | BR1 | BR | I | E | E |
| P0: st [0x400] | BX0 | - | M | I | E |
| P0: st [0x310] | BX1 | BX | I | M | E |

□

## Question 9

There are two cores that perform **LD** and **ST**. Each core has a 512 byte 2-way caches with 32 byte cache lines. All the lines are marked invalid before starting execution. In this question, there is a shared L2 cache and a "BUS" between the L1 caches and the L2 cache. **(2 point)** The L2 cache has 1024 direct mapped 32 bytes cache line.




Update the table indicating the bus messages generated (BR, BX, WB) as a result of each operation, and the resulting cache line state. For the bus message indicate the processor source. (E.g: WB0 to indicate that processor 0's L1 cache did a write back). Follow the MSI protocol.

*Solution.*

| Operation | Bus0 | Bus1 | P0 L1 | P1 L1 | L2 |
|---|---|---|---|---|---|
| P0: ld [0x430] | BR0 | BR | S | I | S |
| P0: ld [0x438] | - | - | S | I | S |
| P1: ld [0x300] | BR1 | BR | I | S | S |
| P0: st [0x300] | BX0 | - | M | I | S |
| P1: ld [0x300] | BR1, WB0 | - | S | S | M |
| P0: ld [0x430] | - | - | S | I | S |
| P1: ld [0x400] | BR1 | BR | I | S | S |
| P0: st [0x400] | BX0 | - | M | I | S |
| P0: st [0x310] | BX1 | - | I | M | M |

□