



NEW YORK INSTITUTE OF TECHNOLOGY

EENG/CSCI 651 M01 – Computer Architecture 1 – Spring 2016
Semester Project
Deadline: Thursday May 5th at 5:44:59 PM EST

Overview

**This project may be completed individually or in groups of up to two students.
Collaboration between more than two students will constitute an integrity violation.**

You are to write (in **C, C++, Java, or Python**) a program that simulates a simple cache. The parameters associated with the memory system are as follows:

- Memory Block Size = Cache Line Size = 16
- Number of Blocks in Memory =16
- Write Through Cache Policy

Each line of the cache has the following components:

Valid Bit	Set	Tag	Data Values				
			Byte 0	Byte 1	Byte 2	...	Byte 7
0 = Empty 1 = Valid	The set number this cache line belongs to	Block Number of Memory Block Copy					

Your memory should be initialized to zeros at the start of your program.

You have a choice of which type of block placement strategy to implement for your project:

- Direct mapped
- 2-way set associative
- Fully associative

Students who are **working individually** should implement **one** of these three block placement strategies, while students **working in a two person group** should implement **two** of the block placement options.

Please submit your assignment via Blackboard. Assignments submitted via other methods will not be considered for grading.

Your submission should include complete instructions for compiling and executing your code. **If we cannot compile or execute your solution code, it will receive a 0.**



NEW YORK INSTITUTE OF TECHNOLOGY

Your program should have the following functions:

void print_cache(cache_line_t *my_cache)

This function prints all current contents of cache. For example:

```
===== CACHE =====
Set = 0 v = 0 tag = -99 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
Set = 0 v = 0 tag = -99 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
Set = 1 v = 0 tag = -99 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
Set = 1 v = 0 tag = -99 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
Set = 2 v = 0 tag = -99 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
Set = 2 v = 0 tag = -99 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
Set = 3 v = 0 tag = -99 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
Set = 3 v = 0 tag = -99 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
```

void print_mem(int num_cache_lines, int cache_line_size)

This function prints all current contents of memory. For example:

```
===== MEM =====
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15
10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25
20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35
30 31 32 234 34 35 36 37 38 39 40 41 42 43 44 45
40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55
50 51 52 53 54 55 56 57 58 59 60 61 62 66 64 65
60 61 62 63 64 65 66 67 68 69 70 71 72 73 74 75
70 71 72 73 74 75 76 77 78 79 80 81 82 83 84 85
80 81 82 83 84 85 86 87 88 89 90 91 92 93 94 95
90 91 92 93 94 95 96 97 98 99 100 101 102 103 104 105
100 101 102 103 104 105 106 107 108 109 110 111 112 113 114 115
110 111 112 113 114 115 116 117 118 119 120 121 122 123 124 125
120 121 122 123 124 125 126 127 128 129 130 131 132 133 134 135
130 131 132 133 134 135 136 137 138 139 140 141 142 143 144 145
140 141 142 143 144 145 146 147 148 149 150 151 152 153 154 155
150 151 152 153 154 155 156 157 158 159 160 161 162 163 164 165
```

void init_cache(cache_line_t my_cache[], int associativity)

This function initializes the cache to the following state:

```
===== CACHE =====
Set = 0 v = 0 tag = -99 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
Set = 0 v = 0 tag = -99 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
Set = 1 v = 0 tag = -99 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
Set = 1 v = 0 tag = -99 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
Set = 2 v = 0 tag = -99 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
Set = 2 v = 0 tag = -99 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
Set = 3 v = 0 tag = -99 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
Set = 3 v = 0 tag = -99 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
```

150 151 152 153 154 155 156 157 158 159 160 161 162 163 164 165



NEW YORK INSTITUTE OF TECHNOLOGY

int map_mem_block_to_cacheline_for_replacement(int block_num, int associativity, cache_line_t my_cache[])

This function receives the memory block number of interest and calculates the cache line number associated with it using the formula for 2-Way Set Associative caches. Please note that internal to this function, you determine the SET NUMBER of the interest. Then within that set you search for the first cache line whose valid bit is 0. This cache line number then will be returned by this function.

If no cache lines with valid bit equal to 0 were found, then randomly select one of the cache lines to be replaced.

void write_to_cache(int block_num, int offset, int value, cache_line_t my_cache[], int associativity)

This function writes a single value to an offset within a given block number

void load_from_mem(int block_num, int block_size, cache_line_t my_cache[], int associativity)

This function loads a block from memory (identified by the block number) into the appropriate cache line (as per the Set associative cache formula). It also sets the valid bit for this cache line to 1, and copies the block number of the memory into the tag field.

void write_through_to_mem(int block_num, int block_size, int associativity, cache_line_t my_cache[])

This function writes an entire cache line to the appropriate block in the memory associated with it.

int is_cache_hit(int block_num, int associativity, cache_line_t my_cache[])

Gets a mem block number and determines if that block is already in cachememory block number has its tag equal to the block number

int calc_number_of_sets(int associativity)

This function calculates the number of sets in this cache based on the associativity given.



NEW YORK INSTITUTE OF TECHNOLOGY

int calc_my_set_number(int cache_line_num, int num_lines, int associativity)
this function calculates the set number that any cache line belongs to

int map_mem_block_to_set(int block_num, int associativity) ss
based on associativity returns the set number in cache mapped to a mem block number

void store_to_mem(int block_num, int offset, int value)
Stores a single value to the given offset of a given block in memory

The main program should have the following flow:

```
init_cache
init mem

while true
{
    Print the number of hits and misses
    print_mem
    print_cache
    Get from the keyboard the command from the CPU on the bus:
        If the command is READ, get the block number, and offset. Then
            if (is_cache_hit)
                increment num_read_hits counter
            else
                increment num_read_misses counter
                load_from_mem
                print the value of the memory location requested
        If the command is WRITE, get the block number, offset, and value. Then
            if (is_cache_hit)
                increment num_write_hits counter
```



NEW YORK INSTITUTE OF TECHNOLOGY

```
    write_to_cache  
    write_thru_to_mem  
else  
increment num_write_misses counter  
load_from_mem  
print the value of the memory location written  
if the command is QUIT, exit the loop  
{
```



NEW YORK INSTITUTE OF TECHNOLOGY

A sample run of the program is shown below for your reference and understanding:

```
$ ./cache.exe 2
2-way Set Associative Map Cache...
Set Associativity of this cache: 2
Number of sets in this cache: 4
===== cache performance =====
    read misses = 0
    read hits = 0
    write misses = 0
    write hits = 0
===== MEM =====
  0   1   2   3   4   5   6   7   8   9   10  11  12  13  14  15
10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25
20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35
30  31  32  33  34  35  36  37  38  39  40  41  42  43  44  45
40  41  42  43  44  45  46  47  48  49  50  51  52  53  54  55
50  51  52  53  54  55  56  57  58  59  60  61  62  63  64  65
60  61  62  63  64  65  66  67  68  69  70  71  72  73  74  75
70  71  72  73  74  75  76  77  78  79  80  81  82  83  84  85
80  81  82  83  84  85  86  87  88  89  90  91  92  93  94  95
90  91  92  93  94  95  96  97  98  99  100 101 102 103 104 105
100 101 102 103 104 105 106 107 108 109 110 111 112 113 114 115
110 111 112 113 114 115 116 117 118 119 120 121 122 123 124 125
120 121 122 123 124 125 126 127 128 129 130 131 132 133 134 135
130 131 132 133 134 135 136 137 138 139 140 141 142 143 144 145
140 141 142 143 144 145 146 147 148 149 150 151 152 153 154 155
150 151 152 153 154 155 156 157 158 159 160 161 162 163 164 165
===== CACHE =====
Set = 0 v = 0 tag = -99      0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
Set = 0 v = 0 tag = -99      0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
Set = 1 v = 0 tag = -99      0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
Set = 1 v = 0 tag = -99      0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
Set = 2 v = 0 tag = -99      0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
Set = 2 v = 0 tag = -99      0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
Set = 3 v = 0 tag = -99      0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
Set = 3 v = 0 tag = -99      0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
```

Enter choice:

- [0] to read from mem
- [1] to write to mem
- [2] to quit

--> 0

enter block number: 3

enter offset: 2

get_cpu_action(): action = READ, block = 3, offset = 2
 cache read miss block [3] offset[2] --> [0]

===== cache performance =====

read misses =	1
read hits =	0
write misses =	0
write hits =	0

MEM															
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25
20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35
30	31	32	33	34	35	36	37	38	39	40	41	42	43	44	45
40	41	42	43	44	45	46	47	48	49	50	51	52	53	54	55
50	51	52	53	54	55	56	57	58	59	60	61	62	63	64	65
60	61	62	63	64	65	66	67	68	69	70	71	72	73	74	75
70	71	72	73	74	75	76	77	78	79	80	81	82	83	84	85
80	81	82	83	84	85	86	87	88	89	90	91	92	93	94	95
90	91	92	93	94	95	96	97	98	99	100	101	102	103	104	105
100	101	102	103	104	105	106	107	108	109	110	111	112	113	114	115
110	111	112	113	114	115	116	117	118	119	120	121	122	123	124	125
120	121	122	123	124	125	126	127	128	129	130	131	132	133	134	135
130	131	132	133	134	135	136	137	138	139	140	141	142	143	144	145
140	141	142	143	144	145	146	147	148	149	150	151	152	153	154	155
150	151	152	153	154	155	156	157	158	159	160	161	162	163	164	165

===== CACHE =====

Set = 0 v = 0 tag = -99	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Set = 0 v = 0 tag = -99	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Set = 1 v = 0 tag = -99	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Set = 1 v = 0 tag = -99	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

```

Set = 2 v = 0    tag = -99          0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
Set = 2 v = 0    tag = -99          0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
Set = 3 v = 1    tag =   3          30 31 32 33 34 35 36 37 38 39 40 41
Set = 3 v = 0    tag = -99          0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
                                         4
Enter choice:
[0]      to read from mem
[1]      to write to mem
[2]      to quit

--> 0
enter block number: 5

enter offset: 1
get_cpu_action(): action = READ, block = 5, offset = 1
cache read miss block [5] offset[1] --> [0]
===== cache performance =====
    read misses =    2
    read hits =     0
    write misses =   0
    write hits =    0
===== MEM =====
  0  1  2  3  4  5  6  7  8  9  10 11 12 13 14 15
10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25
20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35
30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45
40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55
50 51 52 53 54 55 56 57 58 59 60 61 62 63 64 65
60 61 62 63 64 65 66 67 68 69 70 71 72 73 74 75
70 71 72 73 74 75 76 77 78 79 80 81 82 83 84 85
80 81 82 83 84 85 86 87 88 89 90 91 92 93 94 95
90 91 92 93 94 95 96 97 98 99 100 101 102 103 104 105
100 101 102 103 104 105 106 107 108 109 110 111 112 113 114 115
110 111 112 113 114 115 116 117 118 119 120 121 122 123 124 125
120 121 122 123 124 125 126 127 128 129 130 131 132 133 134 135
130 131 132 133 134 135 136 137 138 139 140 141 142 143 144 145
140 141 142 143 144 145 146 147 148 149 150 151 152 153 154 155
150 151 152 153 154 155 156 157 158 159 160 161 162 163 164 165

```

```

===== CACHE =====
Set = 0 v = 0 tag = -99      0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
Set = 0 v = 0 tag = -99      0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
Set = 1 v = 1 tag = 5       50 51 52 53 54 55 56 57 58 59 60 61 62 63 64 65 66
Set = 1 v = 0 tag = -99      0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
Set = 2 v = 0 tag = -99      0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
Set = 2 v = 0 tag = -99      0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
Set = 3 v = 1 tag = 3       30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46
Set = 3 v = 0 tag = -99      0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

Enter choice:
[0]      to read from mem
[1]      to write to mem
[2]      to quit

--> 1
enter block number: 4

enter offset: 1
enter value: 444
get_cpu_action(): action = WRITE, block = 4, offset = 1 value = 444
cache write miss block [4] offset[1] --> [0]
===== cache performance =====
    read misses = 2
    read hits = 0
    write misses = 1
    write hits = 0

===== MEM =====
 0   1   2   3   4   5   6   7   8   9   10  11  12  13  14  15
10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25
20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35
30  31  32  33  34  35  36  37  38  39  40  41  42  43  44  45
40  444 42  43  44  45  46  47  48  49  50  51  52  53  54  55
50  51  52  53  54  55  56  57  58  59  60  61  62  63  64  65
60  61  62  63  64  65  66  67  68  69  70  71  72  73  74  75
70  71  72  73  74  75  76  77  78  79  80  81  82  83  84  85
80  81  82  83  84  85  86  87  88  89  90  91  92  93  94  95
90  91  92  93  94  95  96  97  98  99  100 101 102 103 104 105

```

100	101	102	103	104	105	106	107	108	109	110	111	112	113	114	115
110	111	112	113	114	115	116	117	118	119	120	121	122	123	124	125
120	121	122	123	124	125	126	127	128	129	130	131	132	133	134	135
130	131	132	133	134	135	136	137	138	139	140	141	142	143	144	145
140	141	142	143	144	145	146	147	148	149	150	151	152	153	154	155
150	151	152	153	154	155	156	157	158	159	160	161	162	163	164	165
<hr/> <hr/> ===== CACHE =====															
Set = 0 v = 0 tag = -99	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Set = 0 v = 0 tag = -99	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Set = 1 v = 1 tag = 5	50	51	52	53	54	55	56	57	58	59	60	61	62	63	64
Set = 1 v = 0 tag = -99	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Set = 2 v = 0 tag = -99	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Set = 2 v = 0 tag = -99	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Set = 3 v = 1 tag = 3	30	31	32	33	34	35	36	37	38	39	40	41	42	43	44
Set = 3 v = 0 tag = -99	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
<hr/> <hr/>															
Enter choice:															
[0]	to read from mem														
[1]	to write to mem														
[2]	to quit														
<hr/>															
--> 0															
enter block number: 4															
<hr/>															
enter offset: 1															
get_cpu_action(): action = READ, block = 4, offset = 1															
cache read miss block [4] offset[1] --> [0]															
<hr/> <hr/> ===== cache performance =====															
read misses = 3															
read hits = 0															
write misses = 1															
write hits = 0															
<hr/> <hr/> ===== MEM =====															
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25
20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35
30	31	32	33	34	35	36	37	38	39	40	41	42	43	44	45
40	444	42	43	44	45	46	47	48	49	50	51	52	53	54	55



NEW YORK INSTITUTE OF TECHNOLOGY

50	51	52	53	54	55	56	57	58	59	60	61	62	63	64	65	
60	61	62	63	64	65	66	67	68	69	70	71	72	73	74	75	
70	71	72	73	74	75	76	77	78	79	80	81	82	83	84	85	
80	81	82	83	84	85	86	87	88	89	90	91	92	93	94	95	
90	91	92	93	94	95	96	97	98	99	100	101	102	103	104	105	
100	101	102	103	104	105	106	107	108	109	110	111	112	113	114	115	
110	111	112	113	114	115	116	117	118	119	120	121	122	123	124	125	
120	121	122	123	124	125	126	127	128	129	130	131	132	133	134	135	
130	131	132	133	134	135	136	137	138	139	140	141	142	143	144	145	
140	141	142	143	144	145	146	147	148	149	150	151	152	153	154	155	
150	151	152	153	154	155	156	157	158	159	160	161	162	163	164	165	
<hr/>																
===== CACHE =====																
Set = 0	v = 1	tag = 4		40	444	42	43	44	45	46	47	48	49	50	51	51
Set = 0	v = 0	tag = -99		0	0	0	0	0	0	0	0	0	0	0	0	0
Set = 1	v = 1	tag = 5		50	51	52	53	54	55	56	57	58	59	60	61	61
Set = 1	v = 0	tag = -99		0	0	0	0	0	0	0	0	0	0	0	0	0
Set = 2	v = 0	tag = -99		0	0	0	0	0	0	0	0	0	0	0	0	0
Set = 2	v = 0	tag = -99		0	0	0	0	0	0	0	0	0	0	0	0	0
Set = 3	v = 1	tag = 3		30	31	32	33	34	35	36	37	38	39	40	41	41
Set = 3	v = 0	tag = -99		0	0	0	0	0	0	0	0	0	0	0	0	0

Enter choice:

- [0] to read from mem
- [1] to write to mem
- [2] to quit

--> q
existing main loop



NEW YORK INSTITUTE OF TECHNOLOGY

You must submit an archive file containing your code, a document containing execution and compilation instructions, and a document containing the sample output of your program's execution.

You must submit your solution file through **Blackboard**. No other form of submission is accepted.

Academic Honesty: All work turned in is expected to be your own work. Do not use another student's work or give your work to others. Do not leave your work lying around where other students can copy it. Any violation of academic integrity is subject to a penalty for all students involved by the NYIT Academic Regulations and Procedures.

Late Submission Grading Policy: Late assignments will be accepted one day late with a 10% penalty. There are no exceptions for any reasons