1. Name: Yiduo Ke. Collaboration: Canal Li
2. True or False
   1. False
   2. False
   3. False
   4. False
   5. True
   6. False
3. Cache Performance
   1. D
   2. E
   3. A
4. Caches
   1. Temporal and spatial locality
   2. Bits:
      1. Direct mapped
         1. offset = 4 bits
         2. index = 3 bits
         3. tag = bits
      2. 2-way set associative
         1. Offset = 3 bits
         2. Index = 3 bits
         3. Tag =
      3. 4-way set associative
         1. Offset = 2 bits
         2. Index = 3 bits
         3. Tag =
      4. Fully associative
         1. Offset = bits
         2. Index = 0 bits
         3. Tag = bits
   3. M

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Address | Direct-mapped | 2-way set-ass (LRU) | 4-way set-ass (LRU) | Fully-ass (LRU) |
| 0x1000 | Cold miss | Cold miss | Cold miss | Cold miss |
| 0x1010 | Cold miss | Cold miss | Cold miss | Cold miss |
| 0x1008 | hit | Cold miss | Cold miss | hit |
| 0x2000 | Cold miss | Cold miss | Cold miss | Cold miss |
| 0x3000 | Cold miss | Cold miss | Cold miss | Cold miss |
| 0x4000 | Cold miss | Cold miss | Cold miss | Cold miss |
| 0x3004 | Conflict miss | hit | Cold miss | hit |
| 0x2001 | Conflict miss | Conflict miss | hit | hit |
| 0x0000 | Cold miss | Cold miss | Cold miss | Cold miss |
| 0x1004 | Conflict miss | Conflict miss | Cold miss | hit |
| 0x1011 | hit | hit | hit | hit |

* 1. D

1. TLB Performance Optimization
   1. bits
   2. Because the index bits for the cache lookup would not change during address translation. The last 8 bits would not change because they’re the page offset, and those 8 bits includes the 4 index bits.
   3. 1000 1000
   4. 0000 0000 1
   5. B and C
2. MMU and Virtual Memory
   1. bits; bits
   2. bits; bits
   3. entries; entries; this is a problem because that’s way too big; there’s no space for the computer to carry out actual tasks, it might as well not have an MMU
   4. 2 MB = bytes. The number of entries is . Size of page table. = kB.
   5. Page directory entry length = bits. # of Page directory entries =
   6. 2MB = bytes. entries. MB = million bytes
   7. A page table entry needs to store a physical page number, which is 34 bits long. This requires 8 bytes.
   8. It takes up too much space to translate to physical memory.
   9. You would then need to go through 5 translations to get the physical address, which might require 6 memory loads. That’s time-consuming.
3. Calling Conventions and Assembly code
   1. Using both caller- and callee-saved registers makes a program faster. Callee-saved registers are preserved across multiple function calls. Caller-saved registers are volatile and are used to do calculation when function calls execute.

PROLOGUE:

BODY:

BGE a1, a0, else

JAL x0, EPILOGUE

else:

add a0, x0, a1

EPILOGUE

jalr x0, ra, 0

1. Full System Layout – D, C, E, E, B, D
2. Linkers and Loaders
   1. E (none of the above)
   2. B (data)
   3. D (stack)
   4. E (none of the above)
   5. C (heap)
   6. It doesn’t even compile because requires . The compilation error:

main.c: In function ‘main’:

main.c:6:31: warning: implicit declaration of function ‘malloc’ [-Wimplicit-function-declaration]

int\* small\_array = (int\*) malloc(SMALL\_ARR\_SIZE \* sizeof(int));

^~~~~~

main.c:6:31: warning: incompatible implicit declaration of built-in function ‘malloc’

main.c:6:31: note: include ‘’ or provide a declaration of ‘malloc’

main.c:11:5: error: expected ‘;’ before ‘printf’

printf("%d\n%d\n", big\_arr[0], small\_array[3]);

^~~~~~

If I include then put the semicolon after the line, and correct big\_arr to big\_array, then it prints out:

**12**

**16**

**done running**

1. Exceptional Control Flow
   1. Asynchronous means that the event is not caused by execution of any instruction, but rather an asynchronous signal that an external IO device sends to the processor.
   2. D
   3. B
   4. C
2. Synchronization
   1. No matter in what order the two threads modify , thread A will add at most to and thread B will add at most to , thus won’t get any higher than because it’s initialized to .
   2. only increases, never decreases. If thread B modified it first so that it becomes , then it will either become or . If thread A modified it first so that it becomes , then it will either becomes or .
   3. .
   4. Matching
      1. A,C
      2. A
      3. B
      4. E
      5. A,D