1. Name: Yiduo Ke. Collaboration: Canal Li

2. True or False
    a. False
    b. False
    c. False
    d. False
    e. True
    f. False

3. Cache Performance
    a. D
    b. E
    c. A

4. Caches
    a. Temporal and spatial locality
    b. Bits:
        i. Direct mapped
            1. offset = $\log_2(\text{block size}) = 4$ bits
            2. index = $\log_2(\text{number of sets}) = 3$ bits
            3. tag = $32 - 4 - 3 = 25$ bits
        ii. 2-way set associative
            1. Offset = $\log_2(\frac{\text{block size}}{\text{ways}}) = 3$ bits
            2. Index = $\log_2(\text{number of sets}) = 3$ bits
            3. Tag = $32 - 3 - 3 = 26$
        iii. 4-way set associative
            1. Offset = $\log_2(\frac{\text{block size}}{\text{ways}}) = 2$ bits
            2. Index = $\log_2(\text{number of sets}) = 3$ bits
            3. Tag = $32 - 2 - 3 = 27$
        iv. Fully associative
            1. Offset = 4 bits
            2. Index = 0 bits
            3. Tag = $32 - 4 - 0 = 28$ bits

c. 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

d. D

5. TLB Performance Optimization
    a. $\log_2 256 = 8$ bits
    b. 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.

c. 1000 1000
d. 0000 0000 1
e. B and C

6. MMU and Virtual Memory
    a. 2^{48} bits; 2.8×10^{14} bits
    b. 2^{64} bits; 1.8×10^{19} bits
    c. $\frac{2^{64}\text{bits}}{16\text{kB}} = \frac{2^{64}\text{ bits}}{2^{14}\text{bytes}} = 2^{50}$ entries; 1.4×10^{14} 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

d. 2 MB = 2^{21} bytes. The number of entries is $\frac{2^{21}}{2^{14}} = 2^7$. Size of page table. = $2^7 \times 8 = 2^{10} = 1$kB.

e. Page directory entry length = $\frac{64-14}{2} = 25$ bits. # of Page directory entries = $2^{25} = 3.4 \times 10^7$

f. 2MB = 2^{21} bytes. $\frac{2^{21}}{2^{14}} = 2^7$ entries. $2^7 \times 8 + 8 + 2$ MB = 2.1 million bytes

g. A page table entry needs to store a physical page number, which is 34 bits long. This requires 8 bytes.
h. It takes up too much space to translate to physical memory.
i. You would then need to go through 5 translations to get the physical address, which might require 6 memory loads. That's time-consuming.

7. Calling Conventions and Assembly code
    a. 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.

b.

```
PROLOGUE:

BODY:
BGE a1, a0, else
JAL x0, EPILOGUE

else:
add a0, x0, a1

EPILOGUE
jalr x0, ra, 0
```

8. Full System Layout – D, C, E, E, B, D

9. Linkers and Loaders
    a. E (none of the above)
    b. B (data)
    c. D (stack)
    d. E (none of the above)
    e. C (heap)
    f. It doesn't even compile because malloc requires stdlib. h. The compilation error:

```
main.c: In function ‘main’:
main.c:6:31: warning: implicit declaration of function ‘malloc’ [-Wimpl
icit-function-declaration]
     int* small_array = (int*) malloc(SMALL_ARR_SIZE * sizeof(int));
                               ^~~~~~
main.c:6:31: warning: incompatible implicit declaration of built-in fun
ction ‘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 stdlib. h then put the semicolon after the small_array[0] = 1776 line, and correct big_arr to big_array, then it prints out:

12
16
done running

10. Exceptional Control Flow
    a. 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.
    b. D
    c. B
    d. C

11. Synchronization
    a. No matter in what order the two threads modify x, thread A will add at most 8 to x and thread B will add at most 4 to x, x thus won't get any higher than 12 because it's initialized to 0.
    b. x only increases, never decreases. If thread B modified it first so that it becomes 1, then it will either become 2 or 3. If thread A modified it first so that it becomes 2, then it will either becomes 4 or 3.
    c. $4,6,8,10,12$.

d. Matching
    i. A,C
    ii. A
    iii. B
    iv. E
    v. A,D

vi.