Part 1 (4 *points*)

1. Assume a memory subsystem with 20 address lines. How many bytes and double-words (64-bit) can be stored in it? Give an example of valid addresses (in hexadecimal form) for bytes and double-words stored in the third quarter of its addressing space. Can the address 7422854_{10} express a valid byte or double-word reference for the memory subsystem? Justify <u>all your claims</u> in detail. (2 points)
2. Write the general equation that solves the following problem: What is the speed up that can be achieved when an application is run in a N processor computer system, where the fraction P of the application execution time in a single processor is concurrent and may be run in parallel. Take also into account that the communication overhead among the processors is increased by 0.1% of the single processor execution time every time the numbers of processors is tripled. Why does not the equation portray the law of Amdahl? How does the equation behave when the numbers of processors tend to infinity?

Part 2 (4 *points*)

3. Consider the following code sequence for the pipelined version of MIPS 64 processor described in the lectures, where the equality test on the contents of the registers for the branch instruction is computed in the instruction decode (ID) stage

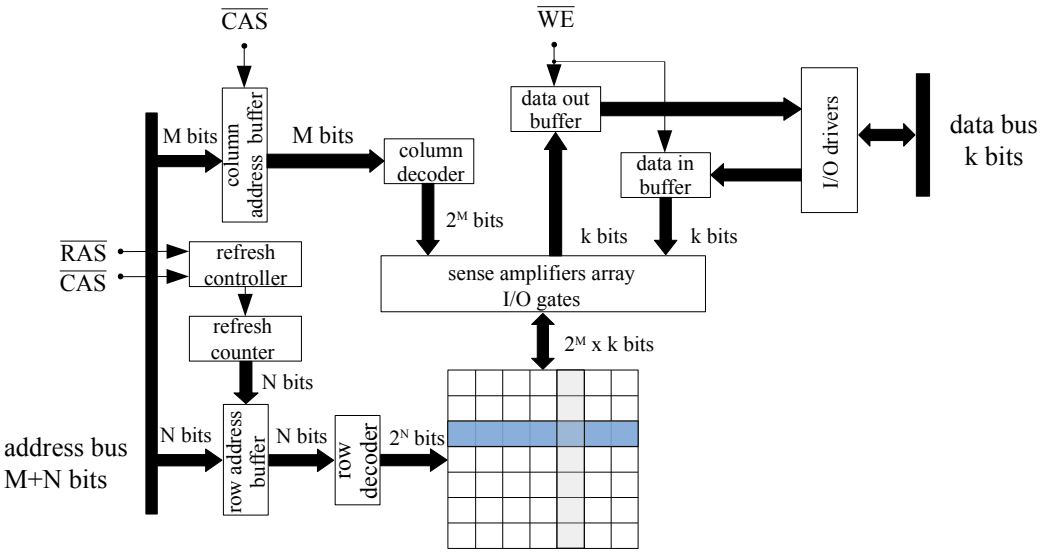
```
        L.D     F2,0(R4)
LOOP:   L.D     F0,0(R1)
        ADD.D   F4,F0,F2
        DADDUI  R1,R1,-8
        S.D     F4,0(R1)
        BNE     R1,R2,LOOP
        MUL.D   F6,F4,F2
```

   Assuming that neither the forwarding, nor the interlocking unit are present, intersperse NOP instructions so that the code can run correctly. What would be the changes on the code that would arise if <u>both</u> units were present? Justify <u>all your claims</u> in detail. (2 points)

4. Assume a 5-stage classical pipeline where multiple functional units are present. (2 points)
   i. What is the reason to employ multiple functional units? Justify <u>all your claims</u> in detail.
   ii. In which pipeline stage are they located? Justify <u>all your claims</u> in detail.
   iii. What is meant in this context by *out of order completion*? Why is this critical in some cases?

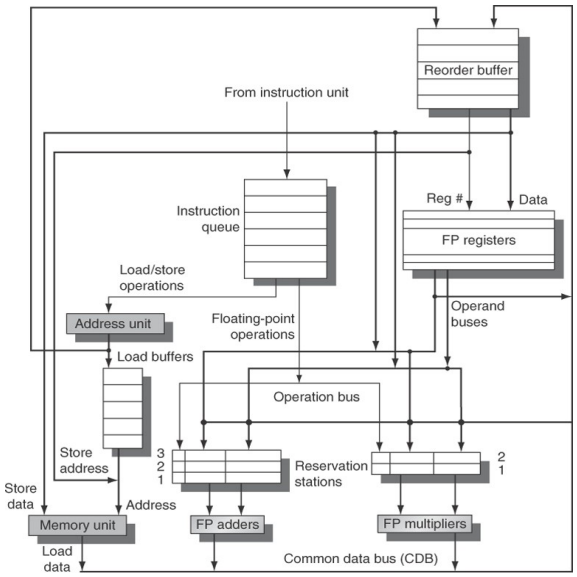
Part 3 (4 *points*)

5. Cache *misses* can be divided into three basic categories. Which are they? Describe them in detail and explain how are they affected by the cache internal organization and size. (2 points)
6. The schematics below depicts the major building blocks of an asynchronous DRAM. Based on it, describe how a *write operation* is carried out. What kind of refreshing takes place at the same time? (2 points)



Part 4 (4 *points*)

7. Sketch in detail the organization of a *tournament branch predictor* which uses a (4,2) correlating branch predictor for global history, keeping track of 4 branch instruction groups, and a (3,2) local branch predictor for local history, keeping track of 8 branch instruction groups. Explain how it works. Assume a 32-bit instruction address. What is the total size in bits of both branch prediction buffers, global and local, and the choice prediction buffer? Justify all your claims in detail. (2 points)
8. The diagram below depicts the basic organization of a floating point unit using the Tomasulo's algorithm extended to handle speculation.



Explain in detail what is the role of the *reorder buffers* in this organization and how exceptions are handled to make them precise. (2 points)

Part 5 (4 *points*)

9. When writing a program in CUDA, there are two optimization steps that must be carried out so that the program runs efficiently in a GPU. Which are they? (2 points)
10. Assume the following CUDA C computation kernel that is run in a 32 X 8 grid of 16 X 32 blocks of threads as the launching configuration.

```
__global__ static void kernel (float *xx, float *yy, float *zz, int N)
{ int x, y, idx;
  x = threadIdx.x + blockDim.x * blockIdx.x;
  y = threadIdx.y + blockDim.y * blockIdx.y;
  idx = blockDim.y * gridDim.y * x + y;
  zz[idx] = xx[idx] * yy[idx % N];
}
```

How many warps are there in total? Which is the separation of the elements of the array `zz` that are computed in the same warp? Justify all your claims in detail. (2 points)