Prepare your answers to the following questions either in a plain text file or in a Microsoft Word file. Answer each question clearly and concisely, but completely, using complete sentences. Explanatory tables and/or diagrams are acceptable, but there must always be a written discussion as well.

Submit your file to the Curator system by the posted deadline for this assignment. No late submissions will be accepted.

The HypoMac machine architecture includes a single processor with the following characteristics:

- a 16-bit data register, called the accumulator (AC)
- a 16-bit instruction register (IR)
- a 12-bit address register, called the program counter (PC), to store the address of the next instruction to be fetched; since there are no branch instructions, the PC is incremented at the end of each fetch-execute cycle
- memory, organized into 4096 16-bit words, addressed from 0 to $2^{12} - 1$
- the first 4 bits of an instruction are an operation code (opcode) uniquely specifying the instruction
- the remaining 12 bits of an instruction specify a memory address or device number, as appropriate
- devices are directly addressable

The machine instructions are:

opcode	meaning	significance of remaining 12 bits
0001	load data into AC	address of data to be loaded
0010	store AC into memory	address to receive data
0011	load data into AC from I/O device	device number
0100	halt	none
0101	add memory contents to AC	address of data to be added to AC
0111	store AC into I/O device	device number

1. [15 points] Given the initial memory and device contents shown below, and the initial processor state shown below, trace the state of the processor registers and memory through each fetch-execute cycle until a halt instruction is reached. Show the state of the processor registers after each fetch-execute cycle and show the final state of memory and devices.

A00	**PC**
??	**AC**
??	**IR**

address	contents
A00	3FF0
A01	5C02
A02	2C01
A03	3FF1
A04	5C01
A05	2C01
A06	7FF2
A07	4C00
A08	2C01
. . .	
C00	0123
C01	011B
C02	05F1
C03	0110
. . .	
FF0	00A0
FF1	0301
FF2	1010

Trace of system registers:

PC	**A00**	**A01**	**A02**	**A03**	**A04**	**A05**	**A06**	**A07**
AC	**00A0**	**0691**	**0691**	**0301**	**0992**	**0992**	**0992**	**0992**
IR	**3FF0**	**5C02**	**2C01**	**3FF1**	**5C02**	**2C01**	**7FF2**	**4C00**

Final memory/device state:

address	contents
A00	**3FF0**
A01	**5C02**
A02	**2C01**
A03	**3FF1**
A04	**5C01**
A05	**2C01**
A06	**7FF2**
A07	**4C00**
A08	**2C01**
. . .	
C00	**0123**
C01	**0992**
C02	**05F1**
C03	**0110**
. . .	
FF0	**00A0**
FF1	**0301**
FF2	**0992**

2. [5 points] In virtually all systems that include DMA modules, DMA access to main memory is given higher priority than processor access to main memory. Explain why this would be preferred.

   If the processor is held up in attempting to read or write memory, usually no damage occurs, except a slight loss of time. But a DMA transfer may be from/to a device that is sending/receiving data in a continuous time-sensitive stream (e.g., a network port, or a tape drive) and cannot be stopped without the loss of data.

3. [6 points] Consider the following code fragment:

```
for ( int i = 0; i < 20; i++ ) {
   for ( int j = 0; j < 10; j++ ) {
      a[i][j] = 0;
   }
}
```

a) Give an example of spatial locality in the code, if there is one.

Spatial locality occurs when the process uses nearby memory contents. This occurs when the array entries are accessed more or less sequentially within the inner loop. One could also argue spatial locality occurs when the instructions in the loop code are executed.

b) Give an example of temporal locality in the code, if there is one.

Temporal locality occurs when the same memory location is used repeatedly over some relatively short time interval. This occurs when the loop index variables `i` and `j` are repeatedly incremented. The repeated accesses to a[i], for a particular value of i, occur within a relatively short time.

4. [9 points] A computer has a cache memory, main memory, and a disk used for virtual memory. When a word is referenced, the time required to access it depends on whether the closest copy of the word exists in the cache, in main memory, or on disk in virtual memory.

Words can only be accessed if they exist in the cache. It takes 20 ns to access a word in the cache. It takes 60 ns to find and copy a word from main memory into the cache. It takes 12 ms, on average, to find a word on disk and copy it into main memory.

90% of references are to words that are already in the cache; 6% of references are to words that are not in the cache but are in main memory; 4% of the references are to words that exist only on disk.

What is the average time (to the closest ns) to access a referenced word on this system?

The average time in nanoseconds, taking into account the probabilities of the three cases, and the times required to copy and access data, would be:

$$t = .90 * 20 + .06 * (60 + 20) + .04 * (12{,}000{,}000 + 60 + 20) = 480{,}026 \text{ ns}$$