Question F6: Structs  [10 pts]

For this question, assume a 64-bit machine and the following C struct definition.

```c
typedef struct {
    char* title;  // title (e.g. "HW SW INTERFACE")
    char dept[3]; // dept (e.g. "CSE")
    short num;   // course number (e.g. 351)
    int enrolled; // students enrolled
} course;
```

(A) How much memory, in bytes, does an instance of `course` use? How many of those bytes are internal fragmentation and external fragmentation? [6 pt]

<table>
<thead>
<tr>
<th>sizeof(course)</th>
<th>Internal</th>
<th>External</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

(B) Assume that an instance `course c` is allocated on the stack and an array `char ar[]` is allocated 40 bytes below `c` (i.e. `&ar + 0x28 == (char*)&c`). Fill in the blanks below with the new ASCII characters stored in `c.dept` after the following loop is executed. Hint: recall that the values 0x30 to 0x39 correspond to the ASCII characters '0' to '9'. [4 pt]

```c
for (int i = 0; i < 52; ++i) {
    ar[i] = i;
}
```

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>___</td>
<td>___</td>
<td>___</td>
</tr>
</tbody>
</table>
Question F7: Caching [19 pts]

We have 256 KiB of RAM and a 4-KiB L1 data cache that is 2-way set associative with 32-byte blocks and random replacement, write-back, and write allocate policies.

(A) Calculate the TIO address breakdown: [3 pt]

<table>
<thead>
<tr>
<th>Tag bits</th>
<th>Index bits</th>
<th>Offset bits</th>
</tr>
</thead>
</table>

(B) The code snippet below accesses two arrays of doubles. Assuming i is stored in a register and the cache starts cold, give the memory access pattern (read or write to which elements/addresses) and compute the miss rate. [6 pt]

```c
#define SIZE 128
double src[SIZE];    // &src = 0x08000 (physical addr)
double dst[SIZE];    // &dst = 0x0E000 (physical addr)
for (int i = 0; i < SIZE; i += 1) {
    dst[i] = src[i];
    src[i] = i;
}
```

<table>
<thead>
<tr>
<th>Per Iteration:</th>
<th>Access 1:</th>
<th>Access 2:</th>
<th>Access 3:</th>
</tr>
</thead>
<tbody>
<tr>
<td>(circle) →</td>
<td>R / W to</td>
<td>R / W to</td>
<td>R / W to</td>
</tr>
<tr>
<td>(fill in) →</td>
<td>_____[i]</td>
<td>_____[i]</td>
<td>_____[i]</td>
</tr>
</tbody>
</table>

Code Miss Rate:

(C) For each of the proposed (independent) changes, draw ↑ for “increased”, ― for “no change”, or ↓ for “decreased” to indicate the effect on the miss rate from Part B for the code above: [8 pt]

- Use float instead  _____
- Half the associativity  _____
- Double the cache size  _____
- No-write allocate  _____

(D) Assume it takes 160 ns to get a block of data from main memory. If our L1 data cache has a hit time of 5 ns and a miss rate of 5%, what is our average memory access time (AMAT)? [2 pt]

ns
Question F8: Processes [18 pts]

(A) The following function prints out four numbers. In the following blanks, list three possible outcomes: [6 pt]

```c
void concurrent(void) {
    int n = 5;
    if (fork()) {
        n++;
        if (fork()) {
            n++;
            wait();
        }
        printf("%d, ", n);
        exit(0);
    } else {
        printf("%d, ", n);
    }
    printf("%d, ", n);
    exit(0);
}
```

(1) ________________  
(2) ________________  
(3) ________________

(B) For the following examples of exception causes, write “S” for synchronous or “A” for asynchronous from the perspective of the user process. [4 pt]

<table>
<thead>
<tr>
<th>System call</th>
<th>Divide by zero</th>
<th>Segmentation fault</th>
<th>Key pressed</th>
</tr>
</thead>
<tbody>
<tr>
<td>____</td>
<td>____</td>
<td>____</td>
<td>____</td>
</tr>
</tbody>
</table>

(C) Fill in the following blanks with “A” for always, “S” for sometimes, and “N” for never if the following would be different when context switching to a different process? [4 pt]

<table>
<thead>
<tr>
<th>Process ID</th>
<th>Program</th>
<th>PTBR</th>
<th>Condition Codes</th>
</tr>
</thead>
<tbody>
<tr>
<td>____</td>
<td>____</td>
<td>____</td>
<td>____</td>
</tr>
</tbody>
</table>

(D) Is the following statement True or False? Provide a brief justification: a single process can execute multiple programs simultaneously. [4 pt]

Circle one: True / False  
Justification:
**Question F9: Virtual Memory [14 pts]**

Our system has the following setup:

- 15-bit virtual addresses and 2 KiB of RAM with 256-byte pages
- A 4-entry fully-associative TLB with LRU replacement
- A PTE contains bits for valid (V), dirty (D), read (R), write (W), and execute (X)

(A) Compute the following values:  [8 pt]

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>page offset width</td>
<td># of TLB sets</td>
</tr>
<tr>
<td># of virtual pages</td>
<td>minimum width of PTBR</td>
</tr>
</tbody>
</table>

(B) Assuming that the TLB is in the state shown (permission bits: 1 = allowed, 0 = disallowed), give example addresses that will fulfill the following scenarios:  [6 pt]

<table>
<thead>
<tr>
<th>TLBT</th>
<th>PPN</th>
<th>Valid</th>
<th>D</th>
<th>R</th>
<th>W</th>
<th>X</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x20</td>
<td>0xc</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0x7f</td>
<td>0xa</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>0x7e</td>
<td>0xf</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>0x04</td>
<td>0xe</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

A value in `%rip` that causes a TLB Hit and no exception: 0x

A *write* address that causes a TLB Hit and segmentation fault: 0x
4. Memory Allocation (11 points total)

```c
#include <stdlib.h>
float pi = 3.14;

int main(int argc, char *argv[]) {
    int year = 2019;
    int* happy = malloc(sizeof(int*));
    happy++;
    free(happy);
    return 0;
}
```

a) [3 pts] Consider the C code shown above. Assume that the `malloc` call succeeds and `happy` and `year` are stored in memory (not in a register). Fill in the following blanks with “<” or “>” or “UNKNOWN” to compare the values returned by the following expressions just before `return 0`.

- `&year` _________ `&main`
- `happy` _________ `&happy`
- `&pi` _________ `happy`

b) [4 pts] The code above has two memory-related errors. Use the line numbers in the code to describe what the errors are and where they occur.

Error #1:

Error #2:

c) [2 pts] (Not related to code at top of page) Give one advantage that next fit placement policy has over a first fit placement policy in an implicit free list implementation.

d) [2 pts] List two reasons why it would be hard to write a garbage collector for the C programming language.

Reason #1:

Reason #2:
5. (11 points) A Nice Hot Cup of Java

WolfBytes has gotten wind of this fancy new language called “Java” and has decide to re-write their website using it. They’ve written two classes to store information about their CPUs:

```java
class CPU {
    float clockSpeed;
    int cacheSize;
    int cacheAssoc;

    int getcores() {
        return 1;
    }
}

class MultiCoreCPU extends CPU {
    int numberOfCores;
    float[] coreSpeeds = new float[16];

    int getcores() {
        return numberOfCores;
    }

    float[] getCoreSpeeds() {
        return coreSpeeds;
    }
}
```

(a) (4 points) The vtable for CPU is shown below. Annotate the diagram with the changes that we would need to make for the vtable of MultiCoreCPU.

You may assume that the alignment for this JVM implementation is the same as C on x86-64, and that fields are stored in memory in the order that they are declared.

(b) (2 points) How much space does an instance of CPU take up?

(b) _______________

(c) (3 points) How much space does an instance of MultiCoreCPU take up?

(c) _______________

(d) (2 points) Give an example of something that is allowed in C, but not in Java, because it would prevent the garbage collector from working properly.