Please read the instructions and questions carefully. You will be graded for clarity and correctness. You have 50 minutes to complete this exam, so focus on those questions whose subject matter you know well. This is a closed note and book exam. Write legibly and check your answers before handing it in.

Short Answer - some will be one or two words – no more than 1 sentence. (40/100 points)

1. (5pts) What information does the global descriptor table hold?

2. (5pts) What C data structure definition allows the program to interpret the same bytes of a variable in different ways?

3. (5pts) What is a string delimiter?

4. (5pts) What is the difference between IO redirection and piping?

5. (5pts) What rights does the group have (and not have) in the rwxr-x-w- policy?

6. (5pts) What does it mean for a memory mapping to be anonymous?

7. (5pts) What is the name of the library that contains most of the implementations for the C programming basic functions
8. (5pts) What does the assert function do?

Long Answer - no more than 4 sentences (20/100 points)

15. (10pts) Identify and explain the difference between the two different interfaces to managing the program break.

16. (10pts) Why can’t you set a watchpoint to a local variable before starting a program?
15. \textit{(10pts)} You have an 8-bit binary number \( x \). You are to assign 4 variables \( a \), \( b \), \( c \), and \( d \) in the following four statements using bit operations. \( a \) should be assigned the value of the top two bits, \( b \) the second two bits, \( c \) the third two bits, and \( d \) the bottom two bits. All assignments should be made as single C statements.

\[
\begin{array}{cccccccc}
\text{bit} & 7 & 6 & 5 & 4 & 3 & 2 & 1 & 0 \\
\text{var} & a & a & b & b & c & c & d & d \\
\end{array}
\]

\begin{enumerate}[(a)]
\item \( a = \) \\
\item \( b = \) \\
\item \( c = \) \\
\item \( d = \)
\end{enumerate}

16. \textit{(10pts)} You are to use \textbf{predefined C string functions}. Assume you have the following strings defined:

```c
char str1 = "this is some text and some more", buf2[100];
char *ptr1, *ptr2;
```

\begin{enumerate}[(a)]
\item In one C statement, assign pointer \( \text{ptr1} \) to the first occurrence of the character \texttt{e} in \texttt{str1}.
\item In one C statement, assign pointer \( \text{ptr2} \) to the last occurrence of the character \texttt{e} in \texttt{str1}.
\item In one C statement, copy the substring of \texttt{str1} from \( \text{ptr1} \) to \( \text{ptr2} \) (inclusively) into \texttt{str2}. Don’t worry about bounds checking or null terminating.
\item In one C statement, append the word "chicken" onto the end of \texttt{str1}. Again, don’t worry about bounds checking.
\end{enumerate}
(a) Create a single declaration of three integer pointers, `valsA`, `valsB`, and `ptr`.

(b) In one C statement, allocate an array of 10 uninitialized ints on the heap and assign its address to `valsA`.

(c) In one C statement, allocate an array of 10 ints that are initialized to zero and assign its address to `valsB`.

(d) In one C statement, set the pointer `ptr` to the address of `valsA[6]` without using any indexing (i.e., no use of [])

```
ptr = 
```

(e) In one C statement, copy the index values of 3, 4, and 5 of `valsA` to index values of 6, 7 and 8 of `valsB`. 
18. (10pts) You have a 4-element fully associative cache that can hold one addressable unit per cache line and an LRU policy. The addressable units have address 0–6, and time is measured using a integer clock. The cache will be asked to processing the following workload and initial cache state:

Workload:
(address 5 at time 11)
(address 6 at time 12)
(address 4 at time 13)
(address 6 at time 14)
(address 0 at time 15)
(address 3 at time 16)
(address 1 at time 17)
(address 2 at time 18)
(address 2 at time 19)

Initial cache state:

<table>
<thead>
<tr>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>Line</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>3</td>
<td>2</td>
<td>5</td>
<td>Address</td>
</tr>
<tr>
<td>8</td>
<td>9</td>
<td>10</td>
<td>3</td>
<td>Last used time</td>
</tr>
</tbody>
</table>

(a) What is the state of the cache after the workload completes?

<table>
<thead>
<tr>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>Line</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Address</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Last used</td>
</tr>
</tbody>
</table>

(b) What is the hit ratio for the workload?

(c) What is the average memory access time for the workload if a cache access cost 100 usec and a miss costs 1000 usec?