Symbol table

Consider the following program implemented in two modules `main.o` and `swap.o`:

```c
/* main.c */
void swap();
int buf[2] = {1, 2};
int main()
{
    swap();
    return 0;
}

/* swap.c */
extern int buf[];
int *bufp0 = &buf[0];
int *bufp1;
int swap()
{
    int temp;
    bufp1 = &buf[1];
    temp = *bufp0;
    *bufp0 = *bufp1;
    *bufp1 = temp;
}
```

For each symbol that is defined or referenced in `swap.o`, indicate whether or not it will have a symbol table entry in the `.symtab` section in module `swap.o`. If so, indicate the module that defines the symbol (`swap.o` or `main.o`), the symbol type (local, global, or extern) and the section (`.text`, `.data`, or `.bss`) it occupies in that module.

<table>
<thead>
<tr>
<th>Symbol</th>
<th><code>swap.o</code> <code>.symtab</code> entry?</th>
<th>Symbol type</th>
<th>Module where defined</th>
<th>Section</th>
</tr>
</thead>
<tbody>
<tr>
<td>buf</td>
<td></td>
<td>global, extern</td>
<td><code>main.o</code></td>
<td>.data</td>
</tr>
<tr>
<td>bufp0</td>
<td></td>
<td>local</td>
<td><code>main.o</code></td>
<td>.text</td>
</tr>
<tr>
<td>bufp1</td>
<td></td>
<td>local</td>
<td><code>main.o</code></td>
<td>.text</td>
</tr>
<tr>
<td>swap</td>
<td></td>
<td>global, extern</td>
<td><code>main.o</code></td>
<td>.data</td>
</tr>
<tr>
<td>temp</td>
<td></td>
<td>local</td>
<td><code>main.o</code></td>
<td>.text</td>
</tr>
</tbody>
</table>
Symbol references

Consider modules that are stored in two files, file 1 and file 2 each. Let $\text{REF}(x.i)$ be a reference to symbol $x$ in file $i$ ($i$ is either 1 or 2), and let $\text{DEF}(x.k)$ be a definition of $x$ in file $k$. We say $\text{REF}(x.i) \rightarrow \text{DEF}(x.k)$ if the linker will associate an arbitrary reference to symbol $x$ in file $i$ with the definition of $x$ in file $k$. For the following modules, how would the linker associate symbols? Write ERROR if there is a link-time error and write UNKNOWN if the linker is free to choose any definition. Assume that symbols used in $\text{REF}(x.i)$ but not explicitly declared in file $i$ are declared $\text{extern}$ in this file to avert compile-time errors.

a) /* file 1 */ /* file 2 */
int x=3; int y;
void main() {
}

\begin{align*}
\text{REF}(x.1) & \rightarrow \text{DEF}(\_\_\_\_\_\_\_\_)\\
\text{REF}(x.2) & \rightarrow \text{DEF}(\_\_\_\_\_\_\_)\\
\text{REF}(y.1) & \rightarrow \text{DEF}(\_\_\_\_\_\_\_)\\
\text{REF}(y.2) & \rightarrow \text{DEF}(\_\_\_\_\_\_\_)
\end{align*}

b) /* file 1 */ /* file 2 */
int x;
int y;
void main() {
}

\begin{align*}
\text{REF}(x.1) & \rightarrow \text{DEF}(\_\_\_\_\_\_\_)\\
\text{REF}(x.2) & \rightarrow \text{DEF}(\_\_\_\_\_\_\_)\\
\text{REF}(y.1) & \rightarrow \text{DEF}(\_\_\_\_\_\_\_)\\
\text{REF}(y.2) & \rightarrow \text{DEF}(\_\_\_\_\_\_\_)
\end{align*}

c) /* file 1 */ /* file 2 */
double x;
void main() {
}

\begin{align*}
\text{REF}(x.1) & \rightarrow \text{DEF}(\_\_\_\_\_\_\_)\\
\text{REF}(x.2) & \rightarrow \text{DEF}(\_\_\_\_\_\_\_)\\
\text{REF}(y.1) & \rightarrow \text{DEF}(\_\_\_\_\_\_\_)\\
\text{REF}(y.2) & \rightarrow \text{DEF}(\_\_\_\_\_\_\_)
\end{align*}

d) /* file 1 */ /* file 2 */
int x;
float y;
void main() {
}

\begin{align*}
\text{REF}(z.1) & \rightarrow \text{DEF}(\_\_\_\_\_\_\_)\\
\text{REF}(z.2) & \rightarrow \text{DEF}(\_\_\_\_\_\_\_)\\
\text{REF}(x.1) & \rightarrow \text{DEF}(\_\_\_\_\_\_\_)\\
\text{REF}(x.2) & \rightarrow \text{DEF}(\_\_\_\_\_\_\_)\\
\text{REF}(y.1) & \rightarrow \text{DEF}(\_\_\_\_\_\_\_)\\
\text{REF}(y.2) & \rightarrow \text{DEF}(\_\_\_\_\_\_\_)
\end{align*}

Relocating Absolute References

The following code shows the object file with relocation entries of the $\text{swap.o}$ module from the previous question.
As you can see, the `swap` function contains five references to be relocated. Upon linking of the `main.o` and `swap.o`, an executable file with the following relocated `.text` and `.data` section is created.

```
08048354 <main>:
  8d 4c 24 04 lea 0x4(%esp),%ecx

08048373:  90 nop

08048374 <swap>:
  55 push %ebp
  89 e5 mov %esp,%ebp
  c7 05 00 00 00 00 04 movl $0x4,0x0
  8b 15 00 00 00 00 mov 0x0,%edx
  a1 04 00 00 00 mov 0x4,%eax

0804956c <buf>:
  01 00 00 00 02 00 00 00

08049574 <bufp0>:
  6c 95 04 08
```

Fill out the table below. For each relocated reference, give (1) the corresponding line number, (2) its run-time memory address, and (3) its value.

<table>
<thead>
<tr>
<th>Line number</th>
<th>Address</th>
<th>Value</th>
</tr>
</thead>
</table>
Relocating PC-Relative References

This problem concerns the disassembly listing of the code below. The purpose here is to give you some practice reading disassembly listings and to check your understanding of PC-relative addressing.

```assembly
-------- Relocated .text section --------
080483b4 <main>:
080483b4: 55 push %ebp
080483b5: 89 e5 mov %esp,%ebp
080483b7: 83 ec 08 sub $0x8, %esp
080483ba: e8 09 00 00 00 call 80483c8 <swap> /* swap(); */
080483bf: 31 c0 xor %eax,%eax
080483c1: 89 ec mov %ebp,%esp
080483c3: 59 pop %ebp
080488c4: c3 ret
080488c5: 90 nop
080488c6: 90 nop
080488c7: 90 nop
080483c8 <swap>:
080483c8: 55 push %ebp
080483c9: 8b 15 5c 94 04 08 mov 0x804945c,%edx /* Get *bufp0 */
080483cf: a1 58 94 04 08 mov 0x8049458,%eax /* Get buf[1] */
080483d4: 89 e5 mov %esp,%ebp
080483d6: c7 05 48 95 04 08 58 movl $0x8049548,0x8049548 /* bufp1 = &buf[1] */
080483dd: 94 04 08
080483e0: 89 ec mov %ebp,%esp
080483e2: 8b 0a mov (%edx),%ecx
080483e4: 89 02 mov %eax,(%edx)
080483e6: a1 48 95 04 08 mov 0x8049548,%eax /* Get *bufp1 */
080483eb: 89 08 mov %ecx,(%eax)
080483ed: 5d pop %ebp
080483ee: c3 ret
-------- Relocated .data section --------
08049454 <buf>:
08049454: 01 00 00 00 02 00 00 00
0804945c <bufp0>:
0804945c: 54 94 04 08 /* Relocated */
```

a) What is the hex address of the relocated reference to swap in line 6?
b) What is the hex value of the relocated reference to swap in line 6?
c) Suppose the linker had decided for some reason to locate the .text section at 0x80483b8 instead of 0x80483b4. What would the hex value of the relocated reference in line 6 be in this case?

**Hand In Instructions**

Hand in your solutions during the exercise class on the due date.