Storage Areas

Storage Organization for a typical procedural language.

```plaintext
--- CODE ---
<p>| STATIC DATA |</p>
<table>
<thead>
<tr>
<th>STACK</th>
</tr>
</thead>
<tbody>
<tr>
<td>HEAP</td>
</tr>
</tbody>
</table>
```

Recursion

```plaintext
void qsort(int m, int n)
{
    int i;

    if (n > m) {
        i = part(m, n);
        qsort(m, i-1);
        qsort(i+1, n);
    }
}
```

Runtime Storage Organization

Storage for code and data,

- **Code Area**: Procedures, functions, methods.
- **Static Data Area**: “Permanent” data with statically known size.
- **Stack**: Temporary data with known lifetime.
- **Heap**: Temporary data with unknown lifetime (dynamically allocated).

Issues in Storage Organization

- **Recursion**
- **Block structure and nesting (nested procedures)**.
- **Parameter passing (by value, reference, name)**.
- **Higher order procedures (procedures as parameters to other procedures)**.
- **Dynamic Storage Management (malloc, free)**.
Organizing Activation Records

Control information for accessing different areas in an activation record:

- **Base Pointer**: Beginning of activation record.
  Arguments are accessed as offsets from base pointer.

- **Environment Pointer**: Pointer to the most recent activation record.
  Usually a fixed offset from base pointer.

- **Stack Pointer**: Top of activation record stack.
  Temporaries are allocated on top of stack.

Activation Records

All information local to a single invocation of a procedure is kept in an **Activation Record**:

- Return Address
- Arguments
- Return Value
- Local variables
- Temporaries
- Other control information
Managing Activation Records

```c
int m(int k)
{
    int i;
    i = k + 15 * n(3);
    return 1(i);
}
```

Managing Activation Records (contd.)

```c
.m:
    pushl %ebp
    movl %esp,%ebp

    . code for m
    movl %ebp, %esp
    popl %ebp
    ret
```