1 Activation Record

The activation record will consist of couple of informations in order for callee to gather all the necessary information to carry out the function/procedure, and for caller to successfully execute and return from the callee.

If it is a function, we will first reserve a space in stack in order to hold to return value. Such memory in the stack allows for callee to store its return value, and it also allows caller to retrieve the return value when the callee is done.

Then, for both function and procedure, we must push the return address to the stack. Such address will be used by the callee to return to the caller when it is done.

Finally, if the function or the procedure contains parameters. It will be included in the activation record as well.

All of the mentioned information must be stored in the activation record in this specific order at all times. For instance, an activation record for `function F( m : integer , n : boolean) : integer` found in "A4-3.488" can be represented as such:

```
1 param : n
2 param : m
3 return address
4 return value
```

Note that in this example, the stack grows up. Hence `param : n` is located on top of the stack. A simpler activation record can be observed with `procedure P` from same sample file:

```
1 return address
```

In this case, only the return address is necessary, since it is a procedure (no return value necessary) without any parameters.

2 Procedure and Function Entrance Code

When the function or a procedure enters, the activation record needs to be properly set up. Please refer to *Section 1* for more information in regards to the information contained within activation record.

Once the activation record has been set up, we must also set the display register of an index equivalent to the caller's lexical level. More information on this can be found in *Section 7*. Now, we can jump to the callee, which effectively allows the program to start the execution of called function.

Let us take `function F(m : integer , n : boolean) : integer` again as an example. We

3 Procedure and Function Exit Code

Do note that in terms of exit code, procedure and functions differs due to the fact that functions must return a value, while procedures do not. The cleanup which caller must do is similar however.

4 Parameter Passing

5 Function Call and Fucntion Value Return

6 Procedure Call

7 Display Management Strategy

When the function references a variable outside of the scope, we must create a reference to the said variable for the correctness of the program. This can be done by scanning through the function and gathering all of the references that were made in the said function. Then, we can add the addreses to the references variables in one of the `display` registers. Then, every time the variable gets references, we can load the address from the corresponding display register.