Computer Architecture - CS 301

Rishit Saiya - 180010027, Assignment - 3

September 19, 2020

1

A recursive call calls itself, either directly or indirectly through another activation block. In direct recursion, a procedure calls itself directly. In indirect recursion, procedure X makes a call to procedure Y, which in turn calls procedure X. The sequence of calls could be longer before a call is made to procedure X.

Each active procedure maintains an activation block, which is stored in the structure of a stack. The activation block, which consists of the parameters such as return address, local variables, and a frame pointer, comes into existence and effect when a call is invoked and closes when the block is terminated.

Thus, for each procedure that is not terminated, an activation block uses part of the stack memory. A stack consists of limited amount of memory and when the limit is reached, the stack overflows. So avoid these crashes, recursion limit is set. The number of activation blocks depend upon the depth of recursion & amount of stack space required to run the program.

2

The main idea of Register Spilling is that instead of filling in the memory to registers for usage, we spill it meaning, intermediate result stored into a temporary memory location. Register Spilling takes place in order to avoid re writing of registers also because registers are limited in most of ISAs. If a register cannot be found for a variable v, we may need to spill a variable. When a variable is spilled, it is stored in memory rather than a register.

The following are the methods of Register Spilling:

- Callee saved: The callee does the storing and restoring of the registers here.
- Caller saved: The caller does the storing and restoring of the registers here.

3

In assembly, all branching is done using following types of instruction:

- ***Conditional Branching:*** In SimpleRISC, there are two conditional branching instructions 'bgt'(branch if greater than) and 'beq'(branch if equal) which check some condition and help us to jump to a particular function we want.
- ***Unconditional Branching:*** In addition to Conditional Branching, for less than, we just use 'b' after 'beq' and 'bgt'. This is also one of the use of unconditional branching. It is followed by cmp instruction which updates the flags register which then used by 'bgt' and 'beq' for branching.
- ***Call:*** They are also useful in doing recursive operations. In SimpleRISC, 'call' instructions are used to call a function. Functions are blocks of assembly instructions that can be repeatedly invoked to perform a certain action. These functions are essential when we need to do a certain set of instructions multiple times, we can call them from anywhere in the program using 'call' instruction.
- ***Return:***'ret' instruction puts the return address into the PC (Program Counter), so that the program can go back to the call instruction.

However, with that being said, its becomes nearly impossible to execute branched, loops or nested loops without some specific instructions. The control flow instructions have different uses and all of them are necessary. Without branching we cannot jump to a different line in a program when need & without call and ret we cannot have functions. So, they are useful for efficient functionality.