Virtual Machine II - Control

Sunday, March 20, 2022 12:35 PM

"If everything seems under control, you're just not going fast enough"
- Mario Andretti (b. 1940), race car champion

Run-time System

Questions to be answered:
How to start a program,
What should the computer do when a program terminates,
How to pass arguments,
How to allocate memory and computing resources to running functions,
How to free memory resources when they are no longer needed.

8.1 High-level Magic

Ex

Function calls:

$x = -b + \sqrt{b^2 - 4 \cdot a \cdot c}$

Can be written as

$x = -b + sqrt(power(b,2) - 4 * a * c)$

Branching:

```
If !(a==0) {x = -b + sqrt(power(b,2) - 4 * a * c)}
Else {x=-c/b}
```

When a caller calls a callee the following must happen:
- Save the return address (address of the caller)
- Save the memory resources of the caller (previous stack)
- Allocate memory resources for the callee's stack
- Make the arguments passed by the caller available to the callee's code (parameters)
- Start executing the callee's code.

When the callee terminates and returns a value the following must happen:

- Make the callee's return value available to the caller's code
- Recycle the memory resources used by the callee
- Reinstate the previously saved memory resources of the caller
- Retrieve the previously saved return address
- Resume executing the callers code, form the return address onward

8.2 Branching

Unconditional branching is specified by the goto instruction:

Goto label

Conditional branching is specified by the if-goto instruction:

If-goto label

Semantics

- *Pop the topmost value off the stack*
- *If true*
- *Jump to label*

8.3 Functions

Lets look at stack addition

Push x
Push y
Add

What if we had a stack operation for power(x, y)=x^y

Push x
Push y
Call power

This would programming and the use of these high-level functions seamless

Calling-chain - all the functions that are currently involved in the program's execution

Return command are specified as:

Redirect the program's execution to the memory location holding the command just following the call command that invoked the current function

1. *Save the return address (caller address + 1) just before control is transferred to executing the caller*
2. *Retrieve the return address and jumping to it just after the callee returns.*

```
//Factorial
int main() {
   return factorial(3);
}

// computes 3!
int factorial( int n ) {
   if ( n==1 ) return 1;
   else return n * factorial( n );
}
```

```
// Tests the factorial function
function main
   push 3
   call factorial
   return
// Computes n!
function factorial(n)
   push n
   push 1
   eq
   if-goto BASE_CASE
   push n
   push n
   push 1
   sub
   call factorial
   call mult
   return
label BASE_CASE
   push 1
   return
```

8.4 VM Specification, Part II

Branching Commands

```
label Label:
   A label is any sequence of letters, digits, underscores(_), dot
   (.), and colons(:) that does not begin with a digit. A label can
   appear before or after it is referenced
goto label:
   unconditional jump.
```

    Find the correct line number -> load A -> 0;jmp
if-goto label:
    Conditional jump.
    pops the top most value of the stack if it is not false then it will jump.

Function Commands

function *functionName nVars:*
    functionName - name of the function
    nVars - The number of local variable
    Instantiate a function.
call *functionName nArgs:*
    functionName - name of the function
    nArgs - The arguments the function passes
    Calling a function
return:
    Transfers the execution to the next command just following the call command in the code of the function that called the current function.

VM Programs

```
   .jack (to be compiled)
-> .vm   (to be translated)
-> .asm  (to be assembled)
-> .hack (binary machine code to be run by the CPU we made)
```

Program entry point

One file in a Jack package must be named Main.jack (with a main() function). This main function will be the entry point of the program.

When you first run a VM program the first function run is argument-less and called Sys.init. This function is a part of the OS and we will go over it more in future chapters. The OS function is programmed to call the Main.jack file and execute the code within.

Program execution

You can either use the VM emulator provided or use the translator we built to run the asm code on the CPU Emulator or use the translated asm file and the Assembler we made to assemble a Hack file, which can then be run on the CPU Emulator (or the Computer we created and loaded into the hardware emulator) again.