Stevens Institute of Technology

Assembly 2 Project

By
Jay Kalyanbhai Savani
20009207

Stevens Institute of Technology

1. Please explain how the stack is used for subroutine call and return

Subroutine refers to repeatedly using the same code. It is often referred to as reusable software or code. The main program can repeatedly jump or call the subroutine. Even though call stack management is crucial for the majority of software to operate as intended, high-level programming languages typically hide and automate the specifics. Special instructions for manipulating stacks are included in many computer instruction sets.

You will recall, the program counter, or register PC (R15), always contains the address of the next instruction to be executed in a program. Calling a subroutine is similar to an unconditional branch as the program jumps to a different address other than the next address. The difference is that when the subroutine finishes executing, control goes back to the instruction after the subroutine call in the main program. A BL instruction causes the address of the next memory instruction to be pushed onto R14 (Link Register or LR) and the argument of BL to be loaded into the program counter. The argument of BL is the starting address of the subroutine. However, you just need to give your program a name while writing it; the assembler will take care of the rest. The BX LR instruction causes the last value stored in LR to be loaded into the program counter at the end of a subroutine. As a result, the instruction in the main program that follows the BL is the one that is performed next. Keep in mind that register LR must be saved and restored on the stack if its value is altered within your function (for example, by executing OutStr or InChar). Using the commands push and pop at the start and conclusion of each subroutine is generally a good habit.

2. Explain a computer's register-level architecture, including: (Maximum 1 point)

A) **CPU-memory interface:** The data bus, address bus, and a few control signals, such as read, write, and memory-function-complete, are all parts of the CPU-Memory interface (MFC). Through the MDR and MAR registers, the CPU is connected to the data bus and address bus, respectively.

   Both fast and slow memories should be handled by the CPU-memory interface circuit. During a certain control step, internal signals produced by the control unit are active for one clock cycle. Slow memory must keep track of the signals for multiple clock cycles. The MFC signal then changes to 1. This is so that the CPU or computer can set them to 1 for a single clock cycle before setting them to 0 and moving on to the next control step. Here, it is expected that the memory is falling-edge triggered or that the MFC signal will change value on the clock's falling-edge. On the rising edge of the clock, the control unit is rising-edge triggered and modifies the value of the control signals.

B) **Special-Use registers:** A special function register, special purpose register, or simply special registers are all considered to be parts of or contained within a microprocessor that handle or control a variety of microprocessor functions.

Depending upon the architecture of processor this could include:
1). Input/output and peripheral control.
2). Timers
3). Stack limit to provide overflows
4). Stack pointers
5). Program counters

Specific registers may not be directly writeable by regular instructions because they are tightly related to a particular special processor function or status.

c) addressing modes: Addressing modes refers to the way in which the operand of the instruction is specified. There are many addressing modes:

1) Instantaneous mode: In this case, the instruction expressly stores the operand as an immediate value. Using SPIM as an example, source.

2) Index Mode: In this case, adding an index register value yields the address of the operand. The instruction code contains both the constant value and the index register number. One illustration is SAL-Accessing Arrays.

3. Indirect Mode: In this case, the operand's actual address is that of a register, a primary memory location, or a location whose address is specified in an instruction or piece of code. Examples are ADD (A), R0.

4) Direct Mode: Also referred to as absolute mode, it embeds the operand's address in the code. For instance, beta: 1000.word.

5) Register Mode: In this case, the operand's value is contained in this mode. The total number of registers from the processor set determines how many bits are needed to specify each register.

Stevens Institute of Technology

References

CS 99s: The Coming Revolution in Computer Architecture. (n.d.). Retrieved November 21, 2022, from

http://cva.stanford.edu/classes/cs99s/

Addressing modes. (n.d.). Retrieved November 21, 2022, from

http://www.cs.iit.edu/~cs561/cs350/addressing/addsclm.html

Computer - USTC. (n.d.). Retrieved November 21, 2022, from

http://home.ustc.edu.cn/~louwenqi/reference_books_tools/Computer-Organization-and-Architecture-9th-Edition-William-Stallings2012.pdf