ASSEMBLY 2 PROJECT

Part 2. Please explain how the stack is used for subroutine call and return

Stacks are an efficient way of storing intermediate data values during complex calculations. Stacks are also an excellent method for storing the return addresses and arguments from subroutine calls.

Program routines that are recursive must “call themselves”. Say a program is in a subroutine and the return address is stored in a fixed location. If the subroutine is called a second time – from inside the subroutine – the address in the fixed location is replaced with the new location. This means that the original return address is lost as it has been replaced. Since the new location is taken from within a subroutine, this will create an infinite loop.

Instead, to prevent this, the return address to be returned is placed in a stack. This time when the routine is again called, the original address is simply pushed down the stack, below the most recent address. This means that the second return address is placed in the stack above the first. When we access the return address we take the address from the top of the stack – the top of the stack is popped. When the next return address is required, it is again taken from the top of the stack. This way, no return address is lost, and every return address is available at the required time.

Part 3. **Explain a computer's register-level architecture, including**

A register is a single, permanent storage location within the CPU used for a particular, defined purpose. A register is used to hold a binary value temporarily for storage, for manipulation, and/or for simple calculations. Registers are basic working components of the CPU.

Registers differ from memory in that they are not addressed as a memory location would be, but instead are manipulated directly by the control unit during the execution of instructions.

a) CPU-memory interface

Two registers, the memory address register and the memory data register, act as an interface between the CPU and memory.

The memory address register (MAR) holds the address in the memory that is to be “opened” for data. The MAR is connected to a decoder that interprets the address and activates a single address line into the memory. There is a separate address line for each row of cells in the memory; thus, if there are n bits of addressing, there will be 2n address lines.

The memory data register (MDR) is designed such that it is effectively connected to every cell in the memory unit. Each bit of the MDR is connected in a column to the corresponding bit of every location in memory. However, the addressing method assures that only a single row of cells is activated at any given time. Thus, the MDR only has access to the values in that single row.

b) special-use registers

Some registers serve many different purposes, while others are designed to perform a single, specialized task. These registers are called special purpose registers. Some important special-use registers are:

- program counter register (PC or IP) holds the address of the current instruction being executed.
- The instruction register (IR) holds the actual instruction being executed currently by the computer.
- The memory address register (MAR) holds the address of a memory location.
- The memory data register (MDR), sometimes known as the memory buffer register, will hold a data value that is being stored to or retrieved from the memory location currently addressed by the memory address register.
- Status registers are several flags grouped together. Flags are 1-bit registers that are used to allow the computer to keep track of special conditions such as arithmetic carry, overflow, power failure, and internal computer error.
- Accumulator register is used for storing the results that are produced by the system.

c) addressing modes

There are alternative ways of extending the addresses specified within instructions so that we can reach more addresses than the size of the instruction address field would, by itself, allow. The different ways of establishing memory addresses within an instruction are called addressing modes.

Different computer architectures vary greatly as to the number of addressing modes they provide in hardware.

References:

Retrieved April 14, 2022, from: Englander, I. (2014). Chapter 7 The CPU and Memory. In *The architecture of computer hardware and systems software: An information technology approach* (5th ed., pp.195–228). essay, Wiley.