Assembly 2 Project

Due on July 7th, 2019
Computer Organization & Programming
CS550WS—Summer I
Ed Banduk

Daniel Kadyrov

Problem 1. Write and submit the Assembly code for "Hello World" program by using the Notepad

Solution

```
START   ORG     $1000

        LEA     MESSAGE,A1
        MOVE.B  #14,D0
        TRAP    #15

    MOVE.B  #9,D0
    TRAP    #15

MESSAGE DC.B    'HELLO WORLD',0


    END START
```

Problem 2. Explain how the stack is used for subroutine call and return

Solution

A stack stores data in a last-in, first-out structure. In other words, the most recent data stored is the first accessed. A call stack uses this data structure to store the return addresses and arguments of subroutine calls. Instead of storing the return address in a fixed location and resulting in an infinite loop, the original address is pushed down the stack allowing the program to rewind in the reverse order the routines were entered. A stack pointer points to the most recent entry at the top of the stack. The bottom of the stack is a fixed memory location.

Englander, I. (2014). *The architecture of computer hardware, systems software, & networking: An information technology approach* (Fifth ed.). Hoboken, NJ: John Wiley & Sons.

Problem 3. Explain a computer's register-level architecture, including:

(a) CPU-Memory Interface

(b) Special-use Registers

(c) Addressing Modes

Solution

The CPU-Memory interface is facilitated through the memory address register and the memory data register. The memory address register, MAR, stores the address of the memory location. A decoder interprets the address and enters an address line into the memory. The memory data register, MDR, holds the data that is being stored or accessed from the address stored in the memory address register. The interaction between the CPU and the memory registers starts with the CPU copying an address from register to the memory address register. The CPU notifies the memory unit whether the transfer is a retrieval from or storage to the memory. The CPU then connects to the MDR and transfers the data from memory into the register or from the register to the memory, depending on the type of transfer. When the data is transferred from the memory into the MDR, the memory data is left intact and the MDR data is replaced with the new data. When the data is transferred from the MDR into the memory, the previous data in the memory is destroyed.

Registers are permanent locations in the CPU that hold binary values for specific functions. They can hold data for processing, instructions for execution, memory address to access, or codes that keep track of the computer status and condition. While general purpose registers store data and addresses, special use registers hold specific information about the program state including the program counter, stack pointer, and the status register. The program counter register holds address of the current instruction being executed. The instruction register holds the instruction that is currently being executed. Status registers hold flags that alert the computer of special conditions like arithmetic carry or overflow, power failure, and other errors.

Addressing modes are different ways of establishing memory addresses in an instruction to be able to reach more addresses than the size of the instruction address field. This is done by storing the address of a memory location as a pointer to a register. Instead of an address field, the instruction provides the register that contains the address. This way, the computer's address capabilities are not limited to and can exceed the size of the address field. The address capabilities of the computer corresponds to the size of the memory address register and can grow with installation of additional memory.

Englander, I. (2014). *The architecture of computer hardware, systems software, & networking: An information technology approach* (Fifth ed.). Hoboken, NJ: John Wiley & Sons.