**Computer System and Concepts**

**Final Project Report**

**Implement Quick Sort Using 8085 Assembly Language**

**Team:**

**Quick Sort Introduction: Chetan Kumar Reddy Vaddi**

**Pin Diagram of 8085: Likitha Korrapati**

* Introduction

**Architecture of 8085: Vivek Reddy Satti**

* Architecture
* Pin Diagram
* Flow of an Instruction cycle

**Implementation of Quick Sort using Assembly Language: - Savata Mohan Venkat Krishna**

* Working
* Coding and Testing
* Result and Conclusion

**Overview:**

Quick Sort is a sorting algorithm which is based on divide and conquer algorithm. This algorithm works by selecting a pivot element from the array and partitioning the other elements or re arrange the array into two sub-arrays such that each element in the left sub-array is less than or equal to the pivot element and each element in right sub-array is larger than the pivot element.

Quicksort is a reliable general purpose sorting algorithm and in this project we can sort an array of elements using the assembly language using quick sort. Quick sort is a widely used sorting algorithm known for its efficiency, with an average time complexity of O(n log n) and can be used for large data sets. The implementation is a good choice for programmers.

In this project, we use Quick Sort algorithm using Assembly Language on a list of arrays in the 8085 Microprocessor which was designed by Intel in 1977. It is an 8-bit microprocessor based on NMOS technology. . It features a 16-bit address bus, an 8-bit data bus, and registers including A, B, C, D, E, H, as well as 16-bit registers SP (Stack Pointer) and PC (Program Counter).

Operating at a clock speed of 3 MHz and supporting a memory addressing range of 64K, the 8085 microprocessors has been used in various applications, including early personal computers, embedded systems, and household appliances like washing machines, microwave ovens, and gadgets.

**Project Scope:**

Understand quick sort concept and working of assemble language. Implement the assembly language code to perform quick sort from an input and produce an output of the result. By doing this we can understand the assembly language instructions and its low level architecture.

**Abstract:**

Quick sort is an efficient sorting algorithm which was developed by British Computer Scientist Tony Hoare in 1959. It is commonly used algorithm for sorting and is faster than other sorting algorithms. The quick sort algorithm arranges a string of numbers or elements in the correct order by using divide and conquer strategy. This algorithm involves selecting a pivot element and partitioning the array into two subarrays.

One subarray of elements less than the pivot and other subarray if elements greater than the pivot. The process recursively continues for these subarray until the entire array is sorted. Quick Sort is widely used for its speed and efficiency, especially with large datasets. Programs written in machine language is usually not understood by the people. Assembly language has English words that can be understood and this is a low level language. This assembly program translates the ASCII keyboard inputs into binary machine code for microprocessor.

The 8085 processor is 8 bit processor launched by intel in 1976 using NMOS technology. The configurations of 8085 microprocessor include data bus 8 bit, +5V voltage supply that operated at 3.2 MHz single segment clock. Using 8085 assembly language for quick sort make system efficient in real-time and allows good control over performance.

It serves as an excellent example for teaching programmers about efficient sorting algorithms. While Quick Sort excels in performance for substantial datasets, it may not be the best choice for very small arrays. The configurations of 8085 microprocessor mainly include data bus-8-bit, address bus-16-bit, program counter-16-bit, stack pointer-16 bit, registers 8-bit, +5V voltage supply, and operates at 3.2 MHz single segment CLK. The applications of 8085 microprocessor are involved in microwave ovens, washing machines, gadgets, etc.

**INTRODUCTION**

The 8085 chip is a 8-bit processor introduced by Intel in 1976. It was a huge improvement over its predecessor, the 8080, and was broadly utilized during the 1980s in different PC frameworks, mini-computers, and other embedded Systems.

The 8085 has a 16-bit address bus and a 8-bit data bus, and that implies it can access up to 64KB of memory. It has seven general-purpose registers, including the accumulator, which is used for arithmetic and logic operations, and six different registers that can be utilized for different purposes.

The instruction set of the 8085 contains 246 guidelines that can that can perform a variety of operations, such as arithmetic, logic, data transfer, and program control. The processor also includes support for interrupts, allowing the system to respond to external events.

Regardless of being a older processor, the 8085 is utilized in certain applications today, especially in legacy systems that have not been upgraded. Its relatively simple architecture and instruction set make it a good choice for small-scale embedded systems where power consumption and cost are primary concerns.

**About 8085:**

In 1976, Intel unveiled the 8085 microprocessor, which is an 8-bit microprocessor. It has a 40-pin dual in-line package (DIP) and is designed to work with a clock frequency of up to 3 MHz. The 8085 has a 16-bit address bus and an 8-bit data bus, and it can address up to 64 KB of memory.

**8085 MICROPROCESSOR PIN DIAGRAM:**

**A diagram of a computer program

Description automatically generated**

The 8085 has several pins on its package, each with a specific function. Some of the important pins and their functions include:

* X1 and X2: These pins are connected to an external crystal oscillator that provides the clock signal for the microprocessor.
* Address bus (A8-A15): These pins are used for data transfer and can address up to 64 KB of memory.
* Address/data bus (AD0-AD7): These pins on the microprocessor are called the address/data bus and are capable of multiplexing, meaning they can be used for both transferring addresses and data.
* Address latch enable (ALE): This pin is an active high signal that indicates when the address is on the address/data bus.
* Status signal (IO/M): This pin indicates if the microprocessor is accessing memory or input/output devices with the address on the address/data bus.
* Status signals (S0-S1): These pins indicate the status of the microprocessor, with different combinations indicating different functions.
* RD: This pin controls the microprocessor's read operation.
* The 8085 has several pins on its package, each with a specific function. Some of the important pins and their functions include:
* WR: This pin controls the microprocessor's write operation.
* READY: This pin indicates if a device is ready to either transfer or accept data.
* HOLD: This pin is used to signal that a device is requesting the use of the address and data bus.
* The 8085 has several pins on its package, each with a specific function. Some of the important pins and their functions include:
* INTR: This pin is an interrupt signal with low priority that can be enabled or disabled by software.
* INTA: This pin is used to acknowledge an interrupt signal.
* RST 5.5, RST 6.5, RST 7.5: These pins are used for maskable restart interrupts.
* TRAP: This is a pin that cannot be disabled and has the highest priority in the 8085 microprocessor.
* RESET IN: This pin is a pin that resets the program counter and other flip-flops in the microprocessor.
* RST (RESET) OUT: This pin is a pin used to reset all devices connected to the microprocessor.
* CLK: This pin can be used to generate a clock signal for other peripherals or digital integrated circuits.
* SID and SOD: These pins facilitate the process of transmitting and receiving data serially. 🡪VSS and VCC: VSS is a ground pin that provides the reference voltage for the chip's internal circuitry, while VCC is a pin that supplies a constant voltage of +5V to power the microprocessor's operations.

**Addressing modes in 8085:**

1.Immediate Addressing Mode :

The source operand in this mode is always data. The instruction will take up 2 bytes for data in the 8 bit range and 3 bytes for data in the 16 bit range

Examples:

ADI 05H (add the immediate 8-bit data 05H to the accumulator A)

LXI SP, 1234H (load the stack pointer with the operand 1234H immediately)

CPI 80H (compare the immediate 8-bit data 80H with the accumulator A)

2.Register Addressing Mode

The registers are operands in register addressing mode and contain the data that needs to be acted on. As a result, the operation is completed inside several CPU registers.

Examples:

MOV B, C (move data from register C to register B)

SUB A (subtract data in accumulator A from itself and store the result in the same accumulator)

DCR C (decrement data of register C by one)

3.Direct Addressing Mode

The memory location where the data that has to be operated on is located is given directly as an operand in the direct addressing mode. The operand is immediately made available by the instruction itself.

Examples:

STA 4000H (memory address 4000H: store the data of accumulator A)

LHLD 5000H (load 16-bit memory location 5000H data into pair of H-L registers)

OUT 50H (Send the information from accumulator A to the output port with address 50H)

4.Register Indirect Addressing Mode :

The data that has to be operated is located inside a memory area that is indirectly identified by a register pair in register indirect addressing mode.

Examples:

MOV A, [HL] (transfer the information at the memory location indicated by the H-L pair to the accumulator)

LDAX D (Move the information of the D-E register to the accumulator)

STAX D (store accumulator information in memory pointed by register pair D-E)

4.Indexed Addressing Mode :

The operand in the indexed addressing mode is a memory location that is determined by adding a constant offset to the data in a base register.

Examples:

MOV A, [BC+10H] (transfer the information of the memory address containing the sum of the contents of the B-C register pair & 10H to the accumulator)

LXI H, 2000H (load the H-L register pair with 2000H)

MOV A, [H+10H] (move the information of the memory location of the sum of contents of H-L register pair & 10H to the accumulator)

5.Relative Addressing Mode :

In this mode, a memory location determined by the program counter's contents plus a constant value serves as the operand.

Examples:

JNZ LABEL (If the zero flag is not set, jump to the LABEL)

MOV R0, #30H (move the immediate 8-bit data 30H to register R0)

ADD A, [PC+10H] (add the information of the memory location of the sum of program counter & 10H to the accumulator)

**QUICK SORT:**

Quicksort is a fast-sorting algorithm that works by splitting a large array of data into smaller sub-arrays. This implies that each iteration works by splitting the input into two components, sorting them, and then recombining them. For big datasets, the technique is highly efficient since its average and best-case complexity is O(n\*logn).

It was created by Tony Hoare in 1961 and remains one of the most effective general-purpose sorting algorithms available today. It works by recursively sorting the sub-lists to either side of a given pivot and dynamically shifting elements inside the list around that pivot.

As a result, the quick sort method can be summarized in three steps:

* Pick: Select an element.
* Divide: Split the problem set, move smaller parts to the left of the pivot and larger items to the right.
* Repeat and combine: Repeat the steps and combine the arrays that have previously been sorted.



Computer programmers use the quicksort algorithm to efficiently sort a list or array of items. Quicksort is a popular and widely used sorting algorithm due to its efficiency and speed Its most common uses for programmers include the following:

* A way to learn basic sorting: Quick sort works as a method for teaching new programmers how to sort data sets because the algorithm is straightforward to understand and implement.
* A methodology for sorting tiny datasets: Because it has to repeatedly cycle through the entire set of elements, comparing only two adjacent items at a time, sort is not optimal for more massive datasets. But it can work well when sorting only a small number of elements.
* A sorting methodology for datasets that are mostly in order already: Finally, some computer scientists and data analysts use the algorithm as a final check for datasets they believe are already in nearly sorted order.

**FLOW CHART:**

A diagram of a process

Description automatically generated

The above flow chart explains the implementation of the quick sort algorithm. The algorithm works by first initializing the program. Then, it fetches an element from the input array. If it is the first element, the algorithm creates a new part for it. Otherwise, it compares the element to the last element of the current part. If the element is smaller, the algorithm inserts the element at the beginning of the part. Otherwise, it inserts the element at the end of the part. The algorithm repeats this process until all of the elements in the input array have been covered. Then, the algorithm merges all the parts into a single sorted output array. The flow chart can be further explained as follows.

**Initialization**

The algorithm initializes the program by setting the following variables:

current\_part: The current part being sorted.

last\_element: The last element in the current part.

all\_elements\_covered: A boolean flag indicating whether all of the elements in the input array have been covered.

**Fetch Element**

The algorithm fetches an element from the input array.

**Is it the first element?**

The algorithm checks to see if the fetched element is the first element in the input array.

**Yes**

If the fetched element is the first element in the input array, the algorithm creates a new part for it by setting the current\_part variable to a new empty array.

**No**

If the fetched element is not the first element in the input array, the algorithm compares it to the last element of the current part.

**Comparison**

The algorithm compares the fetched element to the last element of the current part.

**Smaller**

If the fetched element is smaller than the last element of the current part, the algorithm inserts the fetched element at the beginning of the part by calling the insert\_element() function.

**Greater than or equal**

If the fetched element is greater than or equal to the last element of the current part, the algorithm inserts the fetched element at the end of the part by calling the insert\_element() function.

**Insertion of Element**

The insert\_element() function inserts the specified element into the specified array at the specified index.

**All elements covered?**

The algorithm checks to see if all of the elements in the input array have been covered.

**No**

If not all of the elements in the input array have been covered, the algorithm repeats the Fetch Element, Is it the first element?, Comparison, and Insertion of Element steps.

**Yes**

If all of the elements in the input array have been covered, the algorithm merges all of the parts into a single sorted output array by calling the merge\_parts() function.

**Merge Threads**

The merge\_parts() function merges all of the parts into a single sorted output array.

Once the merge\_parts() function has completed, the algorithm has finished sorting the input array.

**PROGRAM:**

**CODING IN SIM8085**

Quicksort in Intel 8085 assembly assumes an array of data is present in memory starting at address 2000H and the array size is 10 elements.

ORG 0000H ; Start of program

LXI H, 2000H ; Load address of the array

MVI C, 0AH ; Number of elements in the array (10)

CALL QUICKSORT ; Call the quicksort routine

HLT ; Halt the program

QUICKSORT:

PUSH H ; Save the base address of the array on the stack

PUSH D

PUSH E

MOV B, M ; Load the number of elements in B

INX H ; Point to the first element

POP E ; Restore E and D registers

POP D

POP H

LXI SP, 00FFH ; Set the stack pointer to 00FFH

LXI H, 0000H ; Set the low and high indices

LXI D, 0009H

LOOP:

XCHG ; Swap low and high indices

MOV A, M

MOV B, A

MVI A, 00H

CPI B

JNC NO\_SWAP

MOV A, M

MOV C, A

MOV M, B

INX H

MOV A, C

MOV M, A

INX D

NO\_SWAP:

DCR D ; Decrement high index

INR H ; Increment low index

CPI H ; Compare low and high indices

JNZ LOOP ; Loop until they are equal

MOV A, H

MOV B, A

MVI A, 00H

CPI B

JZ DONE

PUSH H ; Save the base address of the array on the stack

PUSH D

PUSH E

CALL QUICKSORT

POP E

POP D

POP H

POP H ; Restore the base address of the array from the stack

POP D

POP E

DCR D

PUSH H

CALL QUICKSORT

POP H

DONE:

RET

**ASSEMBLE CODE:**

|  |  |
| --- | --- |
| 21 40 20 | start:LXI H,2040H |
| 16 00 | MVI D, 00H |
| 4E | MOV C,M |
| 0D | DCR C |
| 23 | INX H |
| 7E | check:MOV A,M |
| 23 | INX H |
| BE | CMP M |
| DA 18 08 | JC nextbyte |
| CA 18 08 | JZ nextbyte |
| 46 | MOV B,M |
| 77 | MOV M,A |
| 2B | DCX H |
| 70 | MOV M,B |
| 23 | INX H |
| 16 01 | MVI D,01H |
| 0D | nextbyte:DCR C |
| C2 08 08 | JNZ check |
| 7A | MOV A,D |
| FE 01 | CPI 01H |
| CA 00 08 | JZ start |
| 76 | HLT |

**Results:**

**A screenshot of a computer

Description automatically generated**

**A screenshot of a computer program

Description automatically generated**

**A screenshot of a computer

Description automatically generated**

**Additional Considerations**

Quicksort can be optimized in several ways. For small arrays, insertion sort is often faster than Quicksort due to its lower overhead. Many implementations and applications move to to insertion sort when the array size drops below a certain level. Additionally, using a hybrid algorithm that combines Quicksort with another sorting algorithm like bubble sort or other divide and conquer algorithms can improve performance.

**Applications of 8085:**

Most of the electronic products and systems that we use today have microprocessors. Microprocessors were first created for small, specialized purposes, but they have now developed into a wide range of devices used in everything from modest household appliances to complex systems.

Microprocessors have been used in a variety of everyday products, including game consoles, DVD players, televisions, car tools, test equipment, light dimmers, smoke detectors, hi-fi audio-visual components, and credit card processing systems. Radar systems, satellite communication systems, and digital telephone sets are a few examples of communication systems that use microprocessors. They are also utilized in several medical devices, including alcometers, ultrasonography machines, ECG (Electrocardiogram), and EEG (Electroencephalogram).

Microprocessors' flexibility and capacity to enhance performance with low engineering and production expense is one of its main benefits. For instance, the Intel 8085 microprocessor can carry out a variety of tasks, such as data transfer, addition, subtraction, branching, and storing, thanks to a set of instructions called MOV, MVI, STA, ADD, DIV, and JMP. The TRS-80 Model 100 line and other personal computers that utilized the 8085 CPU.

There are numerous manufacturers of the CMOS version of the NMOS/HMOS 8085 CPU, and some of them add more commands to increase capability. For several NASA and ESA space physics missions, including CRRES, POLAR, FAST, Cluster, HESSI, and others, the radiation-hardened version of the 8085 microprocessor was used in on-board instrument data processors during the 1990s and 2000s.

The 8085 and the 8085-2 served as the CPUs for the PCA1 range of programmable logic controllers manufactured by the Swiss company SAIA in the 1980s. The 8085 and accompanying hardware were also mounted on an STD Bus style card by Pro-Log Corp., which also included a CPU, RAM, sockets for ROM/EPROM, I/O, and external bus interfaces. The item was a direct rival to Intel's selection of Multibus cards.

With the development of microprocessor devices, there was a huge transition and change in life style of people across multiple industries and domains. These microprocessors are greatly used in daily life because of low cost and less weight and less power.

As this microprocessor have multiple simple instructions like Jump, Move, Add, Sub etc., instructions are executed faster and the programming language is understandable by the device and performs many operations like addition, division, multiplication etc. Furthermore many other operations also be done using these microprocessors. These are used in Engineering Application, Medical Field, Communication and Electronic Chips.

**Engineer:**

These microprocessors are used in traffic management devices, servers and medical equipment to process data or operate like lifts, big machinery and lock systems that have automatic entry and exits.

**Medical Field:**

In medical field the microprocessors are used to pump insulin where this processor regulates the device. It operated multiple functionalities like storage calculations, processing of information that received from sensors etc.

**Communication:**

In this field we use these microprocessors in telephonic industry for communications and enhancing efficiency of the devices like digital telephonic systems, data cables and modems. This processors also used in satellite systems, TV which allowed teleconferencing. Even in airlines and railway registration systems, microprocessors are used and to establish LAN and WAN connections across the systems.

**Electronics:**

In the technology of computers, microprocessors are used to implement various types of system like in microcomputers to the range of supercomputers. In gaming industry, many game instructions are developed using these microprocessors.

In televisions, iPad and other virtual devices also there processors are used to perform complicated instructions and functionalities.

**Conclusion**

Quicksort remains one of the most efficient and widely used sorting algorithms due to its simplicity and speed. Its divide-and-conquer approach, combined with its in-place sorting capability, makes it suitable for large datasets and systems with memory constraints. With a good pivot selection strategy, Quicksort can perform consistently well and is a testament to the enduring relevance of classical algorithms in computer science.

This algorithm is a fast-sorting algorithms that works by splitting a large array into sub array that implies that each iteration works by splitting the input into two components, sorting then and again combining them. This is highly efficient for big datasets which gives best time complexity. This remains the most effective general-purpose algorithms available today.

This works recursively sorting the sub lists to ither side of a given pivot and shifting dynamically elements inside the list around the pivot. As a result, this algorithm selects an element and split the problem set into smaller parts to the left of pivot and larger items to right. And these steps repeat and combine the arrays that have already sorted.

Benefits of Quicksort:

This algorithm works rapidly and effectively and have good time complexity when compared to other algorithms. So, this is an excellent choice for situations when space is limited.

Limitations of Quicksort:

This technique is unstable as it cannot maintain key-value pairs initial order and when the pivot element is very large or very small or all components have same size then the performance of this quicksort is significantly impacted by these worst-case scenarios.

Also, it is difficult to implement as it’s a recursive process if the recursion is not available. Hence, it is efficient if the process have recursive and iterative ways and is in-place, cache-friendly and also a tail-recursive algorithm.

**References:**

<https://www.elprocus.com/8085-microprocessor-architecture>

<https://www.tutorialspoint.com/data_structures_algorithms/quick_sort_algorithm.htm>

<https://www.geeksforgeeks.org/pin-diagram-8085-microprocessor/>

<https://usemynotes.com/8085-pin-diagram-in-microprocessor/#:~:text=The%208085%20pin%20diagram%20consists,and%20serial%20input%2Foutput%20ports>.

[8085 program for quick sort (tutorialspoint.com)](https://www.tutorialspoint.com/8085-program-for-bubble-sort)

<https://www.w3resource.com/csharp-exercises/searching-and-sorting-algorithm/searching-and-sorting-algorithm-exercise-9.php>

8085 Sim online complier: <https://www.sim8085.com/>

<https://www.geeksforgeeks.org/addressing-modes-8085-microprocessor/>

<https://en.wikipedia.org/wiki/Intel_8085>

<https://www.iosrjournals.org/iosr-jece/papers/Vol.%2010%20Issue%206/Version-2/C010620914.pdf>