第 4 回 実験レポート

森田 蓮

2024 年 12 月 27 日

1　目的

本実験の目的は以下の通りである．

1. コンパイラを使わずに，アセンブリ言語・機械語によって整列アルゴリズムを直接記述することが可能であることを示す。
2. 疑似コードや高級言語での実装と比べて，大きく異なる点があるかどうかを明らかにする。また，コードの複雑さやコード量がどの程度増すかを明らかにする。

2　方法

初めにアセンブリ言語で整列アルゴリズムを実装する．次にアセンブリ言語で実装した整列アルゴリズムが適切な動作をするか入力データを与え実行されるか確認する．

また，アセンブリ言語で実装した整列アルゴリズムの疑似コードを高級言語 (java) で実装しコードの複雑さやコード量の変化の比較対象とする．

3　結果

実験の結果，アセンブリ言語で記述した整列アルゴリズムは正しく動作することが確認された．

4　考察

参考文献

浜元信州, 井田寿朗, 齋藤貴英, 酒井秀晃, 小田切貴志, 横山重俊, 動的 VLAN を利用した全学認証ネットワークの構築, 2016 年 20 巻 1 号 p. 65-74 https://www.jstage.jst.go.jp/article/

jacn/20/1/20_JACN20-1-8/_article/-char/ja/

付録

アセンブリ言語:

```
    section .data
    data:  dd 45, 12, 78, 34, 56, 255,
         100, 23    ; 配列データ
    ndata equ 8   ; 配列の要素数
section .text
    global _start

_start:
    mov ecx, data
    mov esi, ndata

Outloop:
    dec esi
    jz end              ;ecx = 0 で終了
    mov edi, ecx
    mov edx, esi

Inloop:
    mov eax, [edi]          ;今の値
    mov ebx, [edi + 4]      ;次

    cmp eax, ebx
    jbe Noswap        ; eax >= ebx
        Noswap

    mov [edi], ebx
    mov [edi + 4], eax

Noswap:
    add edi, 4
    dec edx
    jnz Inloop
    jmp Outloop

end:
    mov eax, 1
    mov ebx, 0
    int 0x80
```

Java コード:

```
public class BubbleSort {
    private int[] sort_array;
    private long timeStart;
    private long timeStop;
    private int assignCount;
    private int compareCount;
    private BubbleSort() {
    }

public BubbleSort(int[] sort_array) {
    this.sort_array = sort_array;
    }

public void swap(int x, int y) {
    assignCount += 3;
    int temp = this.sort_array[x];
    this.sort_array[x] = this.
        sort_array[y];
    this.sort_array[y] = temp;
    }

public void bubbleSort() {
    for (int i = this.sort_array.
        length - 1; i > 0; i--) {
    for (int j = 0; j < i; j++) {
        if (compareValue(sort_array[j
            ], sort_array[j + 1]) ==
            -1) {
            this.swap(j, j + 1);
            }
        }
    }
}
public boolean checkSort(){
    for(int i = 0; i < this.sort_array
        .length - 1; i++){
        if(sort_array[i] > sort_array[
            i + 1]){
            return false;
```

```
        }
    }
    return true;
}
public void startTimer(){
    timeStart = System.nanoTime();
}
public void stopTimer(){
    timeStop = System.nanoTime();
}
public long getSortTime(){
    return (timeStop - timeStart);

}
public int getAssignCount(){
    return assignCount;
}
public int compareValue(int n1, int n2
    ){
    compareCount++;
    if(n1 < n2){
        return 1;
    }else if(n1 > n2){
        return -1;
    }

    return 0;
}
public int getCompareCount(){
    return compareCount;
}
}
```