情報学群実験第 2 レポート

アセンブリ言語による整列アルゴリズムに関する実験

1250373 溝口洸熙 *

2022 年 11 月 1 日

目次

* 高知工科大学 情報学群 2 年 清水研究室

1　アセンブリ言語による整列アルゴリズム記述可否の検証

1.1　実験の目的

高級プログラミング言語，Java，C，Python などは，『コンパイラ』と呼ばれる装置を通して機械語に書き換えられ，コンピュータで実行されている．

それに対して，アセンブリ言語は各機械語命令につけられた「意味する名前」（ニーモニック；mnemonic）を使ってプログラムを表記する表記法である．[1] また，アセンブリ言語表記を機械語のビット列に変換する作業をアセンブルと言い，それを行うソフトウェアをアセンブラと言う．つまり，コンパイラとアセンブラは別物であり，アセンブラは機械語の表記を変えたものである故にコンピュータへの命令を 1 対 1 で書き換えるものである点がコンパイラと大きく違う点である．

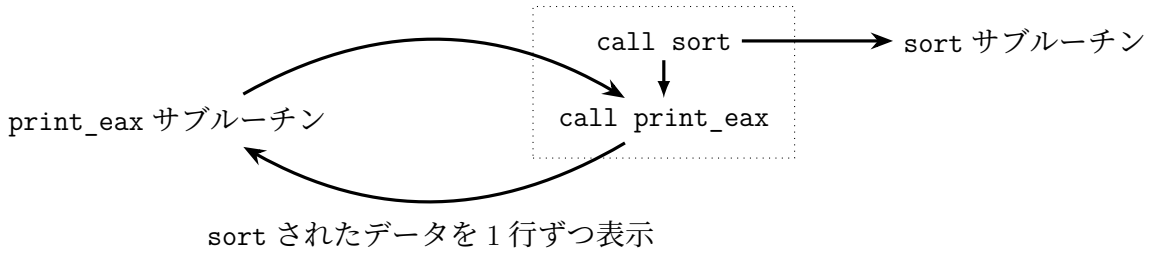
本実験課題の目的は，このようなアセンブリ言語・機械語に対して，コンパイラを使わずに整列アルゴリズムを直接技術することが可能であるか確認することである．

1.2　プログラムの仕様

0 以上 2^{31} 未満の自然数列に対して昇順に整列するアルゴリズムをアセンブリ言語で記述する．その際，test_sort.s ファイルが sort サブルーチンを呼び出して整列を行う．検証したい自然数列は，test_sort.s 内の data1 にダブルワードで，データの個数は ndata1 で定義しており，test_sort.s 実行時 data1 に格納してある自然数列が print_eax サブルーチンによって出力される．

data1 の先頭番地は EBX，ndata1 は ECX に格納する．sort の呼び出し前後で他の汎用レジスタの値は変化しないように設計されている．内部処理の概要を Fig 1.1 に示す．整列アルゴリズムは，**バブルソート**アルゴリズムを元に作成している．

Fig 1.1: 処理概要



1.3　実験

1.3.1　実験方法

以下のコンピュータを利用して，アセンブリ言語で書いた整列プログラムを実行可能ファイルに変換し実行した．

ここで，data1 のデータをどのように設定するかが大切である．例えば，データ列が同じ数値の場合，整列処理の際正しい結果になりうるかの検証をする．

```
$ uname -a
Linux KUT20VLIN-462 5.4.0-70-generic #78~18.04.1-Ubuntu SMP Sat Mar 20 14:10:07 UTC
2021 x86_64 x86_64 x86_64 GNU/Linux
```

print_eax.s，test_sort.s，sort.s をそれぞれアセンブルして実行する．

```
$ nasm -felf print_eax.s ; nasm -felf test_sort.s ; nasm -felf sort.s
$ ld -m elf_i386 print_eax.o test_sort.o sort.o
$ ./a.out
```

実行結果を確認し，整列されているか確認するとともに，Java で記述したプログラムとアセンブリプログラムを比較してどの部分をどのようにアセンブリ言語化したかを確認する．

1.3.2 実験結果

実験結果と期待される結果を Tbl 1.1 に示す．これからわかるように，自然数列を昇順に整列されていることが確認できる．

Tbl 1.1: 実験結果と比較

入力	1 3 5 7 9 2 4 6 8 0 1 2
出力	0 1 1 2 2 3 4 5 6 7 8 9
期待出力	0 1 1 2 2 3 4 5 6 7 8 9

Java で記述したバブルソート（src 1.1）と，アセンブリ言語で記述したバブルソート（src 1.2）を比較して，言語化の箇所を確認する．src 1.1 は入力データ列を data とし，src 1.2 は EBX に入力データ列の先頭番地，ECX 入力データ列の個数を格納している．ソースコードの各処理の内容と，2 つの言語による対応を Tbl 1.2 に記す．

Tbl 1.2: ソースコードの行対応

処理内容	src 1.1	src 1.2
ループ処理 1 の条件比較・変数処理	3	2 - 3, 25, 26
最大値の更新	4	5
最大値インデックスの更新	5	6
ループ処理 2 の条件比較・変数処理	6	7 - 9, 18, 19
最大値の比較・更新	7 - 11	10 - 17
data[max_index] と data[i] の入れ替え	12 - 14	21 - 24

src 1.1: Java

```java
int max_index = 0;
int max = 0;
for (int i = data.length - 1; i > 0; i--) {
  max = data[0];
  max_index = 0;
  for (int j = 1; j <= i; j++) {
      if (data[j] >= max) {
              max = data[j];
              max_index = j;
          }
      }
      int m = data[max_index];
      data[max_index] = data[i];
      data[i] = m;
}
```

注)
src 1.1，src 1.2 いずれも，整列アルゴリズムの部分のみ掲載している．

src 1.2: アセンブリ

```
loop0:
  cmp   ecx,  0
  jle   endp
  mov   edx,  [ebx]
  mov   eax,  0
  mov   edi,  1
  loop1:
    cmp   edi,  ecx
    jg    loop0l
    mov   esi,  [ebx + edi*4]
    cmp   esi,  edx
    jge   then
    jmp   endif
    then:
      mov edx,  [ebx + edi*4]
      mov eax,  edi
    endif:
      inc edi
      jmp loop1
  loop0l:
    mov   esi,  [ebx + eax*4]
    mov   edi,  [ebx + ecx*4]
    mov   [ebx + eax*4],  edi
    mov   [ebx + ecx*4],  esi
    dec   ecx
    jmp   loop0
```

1.4 考察

実験結果より自然数列を整列アルゴリズムをアセンブリ言語で記述できることがわかった．

ただ，これはあくまで 0 以上 2^{31} 未満の自然数に限った整列アルゴリズムであるため，負の整数やその他の有理数などを対象にした整列アルゴリズムが記述可能であるかは，この実験では検証できていない．

2　アセンブリ言語と高級言語の実装に関する違い

2.1　実験の目的

先にも述べたように，アセンブリ言語は機械語を 1 対 1 に記した記法である．それに対して高級言語（高水準言語）は，人間の言語・概念に近づけて設計されてたプログラム言語である．[2]

本実験の目的は，アセンブリ言語や機械語などの低級言語は Java などの高級言語と比べて，実装に関して大きく異なる点があるかどうかを明らかにすることであり，コードの複雑さやコード量がどうなるかを明らかにすることである．

2.2　実験の方法

実装に関して大きく異なる点を明らかにするため，低級言語と高級言語の一般的な実装手順を書き出し比較をする．低級言語はアセンブリ言語，高級言語は Java を利用する．

さらに，コードの複雑さやコード量がどうなるかを明らかにするために，実際に 2 つの言語で書いた同一のアルゴリズムに対して，行数や比較回数を比べる．

2.3　実験の結果

謝辞

本実験課題は本学情報学群 1250372 三上柊氏と共同で実施した．また本学情報学群の高田 喜朗准教授には，整列アルゴリズムに関する様々な助言をいただいた．これらの方々に深く感謝いたします．

参考文献

[1] 高田喜朗. 情報学群実験第 2 テキスト, 第 1 章.
http://www.info.kochi-tech.ac.jp/y-takata/pl2/part1/hellohex.html, 2022.

[2] 新村出. 広辞苑. 岩波書店, 第 6 版, 2008.

src 2.1: `sort.s`

```
1       section .text
2       global  _start
3   _start:
4       mov eax,    0
5       mov ebx,    0
```