

# 基礎ソフト 実験レポート 1

C 過程 S-15 組 1TE20137W 2022/10/19

柳 鷹

## 問 1

(1)

```
MOVE.W (%A0)+, (%A1)+
```

上記の 68000 のアセンブラプログラムは A0 レジスタの中身が示すメモリアドレスの値を A1 レジスタの中身が示すメモリアドレスに転送し、この命令実行後、A0 レジスタ・A1 レジスタそれぞれに（ワードサイズだから）2 加算される。

例えば、はじめの A0 レジスタの値が 0x1000、A1 レジスタの値が 0x2000、0x1000 番地の値が 0x1234、0x2000 番地の値が 0x0000 のとき、0x2000 番地に 0x1234 が転送され、その後 A0 レジスタ・A1 レジスタそれぞれに 2 加算されて A0 レジスタが 0x1002 に、A1 レジスタに 0x2002 になる。

(2)

```
.equ TOP, 0xFFFFC00
.equ MASK, TOP+0x80
MOVE.W #0x07F7, MASK
```

上記の 68000 のアセンブラプログラムは、まず上 2 行でシンボルとして TOP を 0xFFFFC00 に、MASK を TOP に 0x80 を加えた値すなわち 0xFFFFC80 に設定する。そして、「MOVE.W #0x07F7, MASK」でシンボル MASK を 0x7F7（ワードサイズ）に再設定する。

(3)

```
.dc.b 'a','b','c','d','e',0
```

上記の 68000 のアセンブラプログラムは、データ列'a','b','c','d','e',0 をバイトサイズでメモリ内に配置する。データが文字列の場合は、ASCII コード（例えば、'a'なら 0x61、'b'なら 0x62）に変換される。

(4)

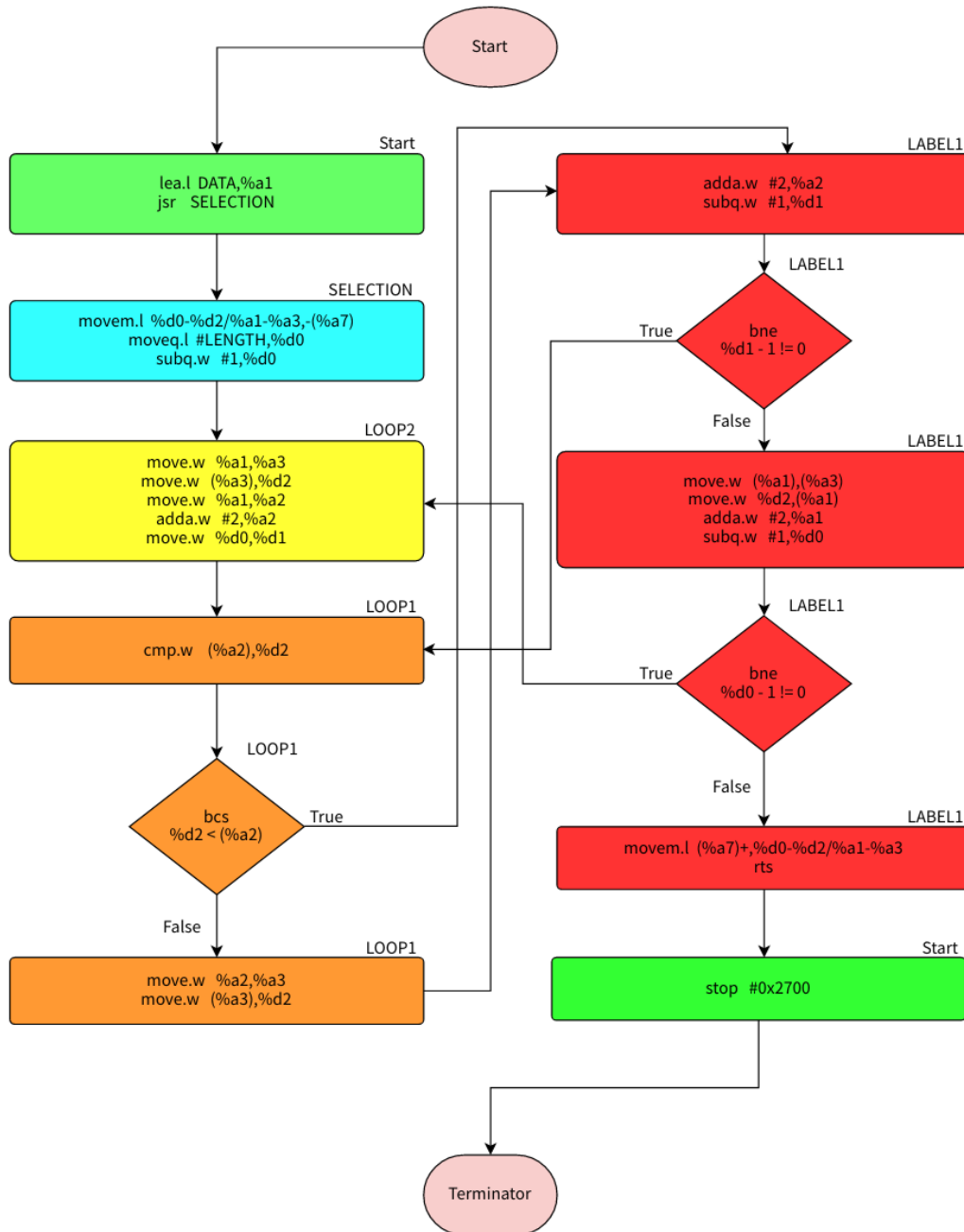
```
.equ BOTTOM,    4  
    MOVE.L %A2,BOTTOM(%A0)
```

上記の 68000 のアセンブラプログラムは、「.equ BOTTOM, 4」でシンボルとして BOTTOM を 0xFFFC00 に設定し、「MOVE.L %A2,BOTTOM(%A0)」で[BOTTOM の中身の値]+[A0 レジスタの中身]となるメモリ番地に A2 レジスタの値を転送する。

問 2

(1)

求めるフローチャートは以下である。



(2)

選択ソートのアセンブラプログラムの実行結果は以下である。

The screenshot displays the BSVC (Basic System V) environment with three main windows:

- Registers:** Shows the state of various registers. A1 is highlighted with the value 0000044c.
- Memory Viewer:** Displays memory contents starting from address 000400. Address 00044c contains the value 0009.
- Program Listing - asc.LIS:** Shows the assembly code. The selection sort algorithm is implemented, with comments in Japanese explaining the steps like finding the minimum element and swapping it.

上の実行結果からこのプログラムが正しく動いていることを簡単に説明する。

まず、整列するデータ列の先頭アドレスを格納している A1 レジスタは 0x00044c が入っており、メモリの 0x00044c 番地からワードサイズ (16 進数 4 桁) でデータが昇順ソートされているのが確認できる。プログラムの実行前はメモリの 0x00044c~0x00044d には 0009 が格納してあった。

(3)

本項での感想・意見を述べる。

本実験までアセンブラプログラムというものにほとんど触れたことがなかった。ただ、最近パチンコやパチスロのメインプログラミング言語がアセンブリ言語であることを知った。それは規則で ROM が最大 16KB や RAM が最大 1024B とメモリの記憶容量が定められているからであった。そしてこの基礎ソフト実験でメモリを意識したアセンブリ言語を体験することによりその理由がよく分かった。本項の選択ソートを実装・実行して感じたのは、メモリ内のデータ配置を把握することができるということであった。アセンブラプログラムは遊技機のようにメモリが限られており、扱うデータ量がそこまで大きくないとき、ソートにかかる計算量よりもメモリの使用について深く考えられるため、適していると感じる。また、アルゴリズムの講義等で計算量について考察する場面はあったが、メモリ等についてあまり考えていなかったように感じたので、この機会に再度学習したいと思った。





問3

(1)

このプログラムでの、メモリ内のデータ配置は以下である。

BSVC: Memory Viewer

File Edit View

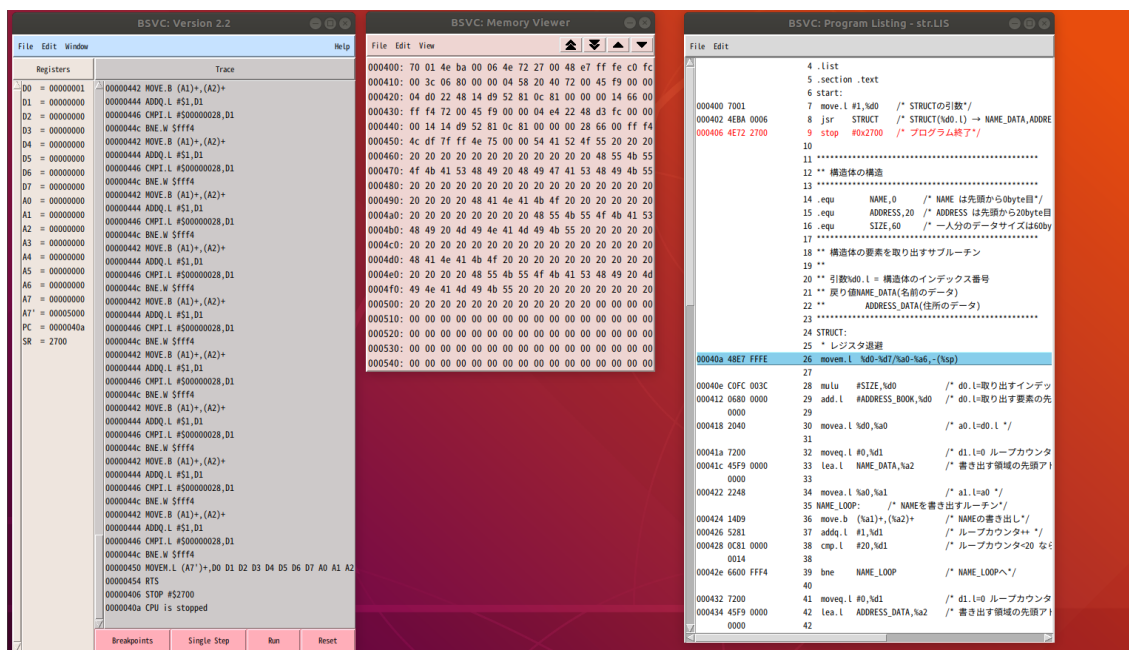


000400:	70	01	4e	ba	00	06	4e	72	27	00	48	e7	ff	fe	c0	fc
000410:	00	3c	06	80	00	00	04	58	20	40	72	00	45	f9	00	00
000420:	04	d0	22	48	14	d9	52	81	0c	81	00	00	00	14	66	00
000430:	ff	f4	72	00	45	f9	00	00	04	e4	22	48	d3	fc	00	00
000440:	00	14	14	d9	52	81	0c	81	00	00	00	28	66	00	ff	f4
000450:	4c	df	7f	ff	4e	75	00	00	54	41	52	4f	55	20	20	20
000460:	20	20	20	20	20	20	20	20	20	20	20	20	48	55	4b	55
000470:	4f	4b	41	53	48	49	20	48	49	47	41	53	48	49	4b	55
000480:	20	20	20	20	20	20	20	20	20	20	20	20	20	20	20	20
000490:	20	20	20	20	48	41	4e	41	4b	4f	20	20	20	20	20	20
0004a0:	20	20	20	20	20	20	20	20	48	55	4b	55	4f	4b	41	53
0004b0:	48	49	20	4d	49	4e	41	4d	49	4b	55	20	20	20	20	20
0004c0:	20	20	20	20	20	20	20	20	20	20	20	20	20	20	20	20
0004d0:	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
0004e0:	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
0004f0:	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
000500:	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
000510:	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
000520:	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
000530:	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
000540:	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00

プログラム実行前のメモリ内のデータ配置を簡単に考察すると、メモリの 0x000400 番地から 0x000455 まではテキスト領域で、0x000458 番地から 0x0004cf までデータ領域である。

(2)

構造体の要素を取り出すアセンブラプログラムの実行結果は以下である。



上の実行結果からこのプログラムが正しく動いていることを簡単に説明する。

STRUCT の出力として、NAME データが 0x0004d0 番地から 0x0004e3 番地に、ADDRESS データが 0x0004e4 番地から 0x0004eb 番地に出力されている。例えば、0x0004d0 番地の 48 は"H"を示している。

(3)

本項での感想・意見を述べる。

本項では、サブルーチンプログラムや A1・A2 レジスタで用いたポストインクリメント付きアドレスレジスタ間接形式について模範コードをトレースしながら学べた。また、C 言語等で聞いたことはあるがあまり意識していなかったメモリのテキスト領域とデータ領域なども Memory Viewer などで視覚的に認識しながら学ぶことが出来てよかった。