情報学群実験第 2 レポート

ARM アセンブリ言語における即値代入の制限に関する実験

1250373 溝口洸熙[*1]

2022 年 12 月 4 日

[*1] 高知工科大学 情報学群 2 年 清水研究室

目次

ソースコード

第 1 章

即値代入の制限における規則性

1.1　実験の目的

arm Assembler の mov 命令[*1]は，i386 Assembler で扱った mov 命令と対応する．しかし，arm Assembler の mov 命令において，レジスタに代入できる即値には制限がある．本実験の目的は，制限について調べることである．

1.2　実験の方法

コンピュータ（src 1.1）を利用して，mov.s（src 1.3）ファイルをアセンブルする．

src 1.1　利用コンピュータ及び実行環境

```
$ uname -a
Linux KUT20VLIN-322 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
$ arm-none-eabi-as --version
GNU assembler (2,27-9ubuntu1+9) 2.27
$ arm-none-eabi-ld --version
 - 不明 -
```

src 1.2　アセンブル

```
1 $ arm-none-eabi-as mov.s -o mov.o
2 $ arm-none-eabi-ld mov.o -o mov
3 $ ./mov ; echo $?
4 3
```

src 1.3　mov.s

```
1     .section    .text
2     .global     _start
3 _start:
4     mov r7, #1
5     mov r0, #3 @ test number
6     swi #0
```

src 1.2: 4 行目は，src 1.3: 5 行目のテスト値を出力する．

アセンブル（src 1.2）の際に src 1.4 の出力が得られた場合は，src 1.3: 5 行目に問題があり即値の代入に失敗している．

src 1.4　Error 出力

```
mov.s: Assembler messages:
mov.s:5: Error: invalid constant (4d1)
   after fixup
```

[*1] レジスタに即値，またはレジスタの値を代入する命令．

様々なテスト値を簡単に試すためにスクリプトファイルによる自動実行プログラム（src 1.7）を作成し，実行した（src 1.6）．実行するに際して，空ファイル `test.s` を作成し，`mov.s` の一部を，変更する必要がある．テストする即値は，$1 \leq N \leq 2^{21}$ である．

src 1.5　`mov.s` 変更後

```
1      .equ     N,   %d
2      .section     .text
3      .global      _start
4  _start:
5      mov r7, #1
6      mov r0, #N @ test number
7      swi #0
```

src 1.6　実行

```
$ bash test.sh
```

src 1.7　`test.sh`

```
1  #!/bin/bash
2  W=0
3  for i in 'seq 0 2097152'
4  do
5  sed -e "s/%d/$i/g" mov.s > test.s
6  arm-none-eabi-as test.s -o test.o > /dev/null 2>&1
7  if [ $? -eq 0]; then
8      echo "$i, $(( $i - $W ))"
9      V=$i
10     W-$i
11 fi
12 done
```

実行結果には，アセンブルできた即値とその即値の前にアセンブルできた即値との差（前項との差）が出力される．

1.3　実験結果

実験結果を Tbl 1.1 に示す．

Tbl 1.1 の他にも，$2^{32}-1$，の値もアセンブルできた．

Tbl 1.1　実験結果

入力数値	アセンブルの可否	前項との差
0	OK	-
1	OK	1
2	OK	1
⋮	⋮	⋮
256	OK	1
260	OK	4
⋮	⋮	⋮
1024	OK	4
1040	OK	16
⋮	⋮	⋮
4096	OK	16
4160	OK	64
⋮	⋮	⋮
16384	OK	64
16640	OK	256
⋮	⋮	⋮
66536	OK	256
65569	OK	1024
⋮	⋮	⋮