



アセンブリ言語

使用するパソコン環境について

情報工学系
権藤克彦



この講義資料は Intel Mac 向け

- 前提とする環境
 - CPU: Intel x86-64
 - OS: macOS
 - 言語処理系: gcc (clang)
- アセンブリ言語は環境（プラットフォーム）に強く依存
 - gcc -S で出力したコードが、環境ごとに微妙に異なる
 - 自分が使用する環境は慎重に選択する



gcc -S の結果：Intel Mac

```
int add5 (int n)
{
    return n + 5;
}
```

foo.c

- x86-64 のアセンブリコード

```
% gcc -S foo.c
```

1行目は .text でも代用可

```
.section    __TEXT,__text,regular,pure_instructions
.p2align   4, 0x90
_add5:
    pushq %rbp
    movq %rsp, %rbp
    movl %edi, -4(%rbp)
    movl -4(%rbp), %eax
    addl $5, %eax
    popq %rbp
    retq
.subsections_via_symbols
```

foo.s

コメントや .cfi関連のアセンブラ命令を削除



gcc -S の結果：M1 Mac

- ARM のアセンブリコード

```
.section __TEXT,__text,regular,pure_instructions
.globl _add5
.p2align 2
_add5:
    sub sp, sp, #16
    str w0, [sp, #12]
    ldr w8, [sp, #12]
    add w0, w8, #5
    add sp, sp, #16
    ret
.subsections_via_symbols
```

出力内容がぜんぜん違う！



gcc -S の結果：M1 Mac

- Rosetta2 を使うと，Intel Macと同じ結果になる
- M1 Macの人は **Rosetta2 使用を強く推奨**

```
.section      __TEXT,__text,regular,pure_instructions
.p2align     4, 0x90
_add5:
    pushq %rbp
    movq %rsp, %rbp
    movl %edi, -4(%rbp)
    movl -4(%rbp), %eax
    addl $5, %eax
    popq %rbp
    retq
.subsections_via_symbols
```



VSCodeもRosetta使用がオススメ

Rosetta2 の使い方

初回起動時にRosettaのインストールを求められる

1. Finderでアプリケーションの下のユーティリティを開く
2. ターミナルを右クリックし「情報を見る」を選択
3. 「Rosettaを使用して開く」をチェック
4. ターミナルを起動





Rosetta2 とは？

- x86-64の実行可能ファイルをARM上で実行するエミュレータ
 - x86-64コードを動的にARMに変換して実行
- MacのUniversalバイナリ
 - 複数のアーキテクチャ用コードを含んでいる
- 高速に動作し、互換性も高い
 - 多くのエミュレータは遅いし、完全互換ではない

要するに M1 Mac上で、Intel Macのアプリの実行を可能にする技術



gcc -S の結果：Linux (Ubuntu)

- Linux (Ubuntu-20.04 LTS) @ VMWare

```
.file    "foo.c"
.text
.globl   add5
.type    add5, @function

add5:
    endbr64 # ROPを防ぐ仕組み。通常は nop と同じ。
    pushq   %rbp
    movq    %rsp, %rbp
    movl    %edi, -4(%rbp)
    movl    -4(%rbp), %eax
    addl    $5, %eax
    popq    %rbp
    ret
                                                .note.gnu.section などは削除
    .size   add5, .-add5
    .ident  "GCC: (Ubuntu 9.3.0-17ubuntu1~20.04) 9.3.0"
    .section .note.GNU-stack,"",@progbits
```




gcc -S の結果：Linux (Ubuntu)

- Linux (Ubuntu-20.04 LTS) @ WSL2

```
.file    "foo.c"
.text
.globl   add5
.type    add5, @function

add5:

    endbr64
    pushq   %rbp
    movq    %rsp, %rbp
    movl    %edi, -4(%rbp)
    movl    -4(%rbp), %eax
    addl    $5, %eax
    popq    %rbp
    ret

.size     add5, .-add5
.ident    "GCC: (Ubuntu 9.3.0-17ubuntu1~20.04) 9.3.0"
.section   .note.GNU-stack,"",@progbits
```



64ビットの

Windows PCの人は Linux環境を準備

- Linux は以下の選択肢あり
 - WSL2 (Windows Subsystem for Linux 2)
 - <https://www.kkaneko.jp/tools/wsl/wsl2.html> ← 権藤はここを見た
 - VirtualBoxなどの仮想マシンソフト上に Ubuntu などをインストール
 - Docker for Windowsをインストールし，その上で Ubuntu などを使う
 - オンライン環境 <https://repl.it/> 上でプログラミングする
 - お手軽だが，lldbデバッガを使えず不便. gdbは使える.
- Windows の Cygwin や MinGW は強く非推奨
 - UNIXとしての互換性が低いから
- デバッガも準備
 - lldb か gdb を（インストール方法はググってね）



情報工学系計算機室（CSC）の使用

- 使用OK
- リモートログインは不可