# 早稲田大学

# コンピュータアーキテクチャ A 実習用アセンブラ tas 説明書

木村啓二

2016 年 10 月

（2019 年 9 月: nor, andi 命令を追記）

## 1. はじめに

本文章はコンピュータアーキテクチャAの実習で用いる簡易CPU シミュレータ(tmips)用のアセンブラであるtasの説明書である。本アセンブラはPythonで記述された極めてシンプルなものであり、改変は容易である。
なお、本アセンブラの入力部は極めて簡易的なものである。コンパイラやアセンブラの入力ファイルを扱う一般的な方法は3年生設置科目「言語処理系」で取り扱う。

## 2. アセンブラの実行

ソースプログラムのファイル名を指定してアセンブラのコマンドtasを実行する。ソースプログラム“tas.s”のアセンブルは以下の様になる：

```
tas sample.s
```

アセンブル結果のtmips機械語が記述されたテキストファイルをsuffixを.datに置き換えたファイルとして出力する。命令は0番地から順に配置される。これをmemfile.datと名前を変更しtmipsのverilogファイルのあるディレクトリにコピーする。testbench.vのテストコードも適宜修正する。その後、iverilogによるコンパイルとvvpによるシミュレーションを実行する。

```
iverilog -s testbench -o testbench *.v
vvp testbench
```

top.vに記述されている“memfile.dat”をアセンブラが出力したファイル名としても良い。

## 3. アセンブラの文法

### 3.1. 定数

アセンブラのソースファイル中で定数を利用する場合、整数のみ使用することができる。整定数の表記方法として、10進数と16進数を利用できる。16進数の場合は先頭に“0x”をつける。10進数の場合、先頭に“-”をつけて負数を表すこともできる。

例： 1234, -5678, 0xabcd

### 3.2. レジスタ

命令のオペランドにレジスタを利用する場合、2種類の表記方法を利用できる。
一つは“$”に続けてレジスタ番号を整数値で記す方法であり、レジスタ番号を0から31の間で利用可能である。

例： $0, $3, $5

もう一つは“$”に続けてレジスタの種別を表す名前を続ける方法である。

例：$zero, $t0, $sp

両者の対応関係は以下の通りである：

- $zero / $0
- $at / $1
- $v0-$v1 / $2-$3
- $a0-$a3 / $4-$7
- $t0-$t7 / $8-$15

- $s0-$s7 / $16-$23
- $t8-$t9 / $24-$25
- $k0-$k1 / $26-$27
- $gp / $28
- $sp / $29
- $fa / $30
- $ra / $31

### 3.3. メモリオペランド

ロード命令やストア命令でアドレスを指定するために用いる。以下のフォーマットである。

“定数”(レジスタ)

レジスタ中の値に“定数”で指定された値を足し、結果をメモリ中のアドレスとして利用する。アドレスは 4 の倍数でなければならない。そうでない場合の結果は不定である。“定数”を 10 進数で表記する場合は負の数も扱える。“定数”は 16 ビットで表現できる値でなければならない。16 ビットを超えた場合の動作は不定である。

例： 4($t0), -8($t1)

### 3.4. ラベル

アドレスを表す定数の代わりにラベルを利用することができる。ラベルは英字アルファベットで始まり英字アルファベットか数字の続きで表記される識別子である。

例： label, F001

### 3.5. 演算子

演算子として hi()と lo()を使える。hi()は整定数あるいはラベルの値の上位 16 ビット、lo()は下位 16 ビットを抽出する。演算子の入れ子は許されない。

例： hi(0x12345678) (演算結果は 0x1234)

lo(F001) (結果は F001 と対応するアドレスの下位 16 ビット)

### 3.6. コメント

ソースファイル中、“;”以降はコメントとして扱われる。すなわち、アセンブラ内部では無視される。

### 3.7. 1 行の構成

ソースファイルの 1 行の構成は以下の通りである：

[ラベル:][命令]

“[]”は省略可能であることを表す。

“ラベル:”により、“ラベル”に記載される識別子にその行に対応するアドレスを代入される。アドレスは MIPS 命令ごとに 4 バイト増加する。

“命令”は疑似命令もしくは MIPS 命令を記載する。

例：

```
            add $t0,$t1,$t2
label1:  lw $s0,-4($t0)
            .dw  1234
```

### 3.8. 疑似命令

ソースファイル中でアセンブラに指示を与える命令を疑似命令と呼ぶ。tas では疑似命令として.dw を利用できる。.dw はその行に対応するアドレスで示された位置に指定された値を配置する。

例：

```
            .dw 0xabcd
label2:  .dw 1234   (MIPS 命令から値の配置されたアドレスを label2 で参照できる)
```

### 3.9. サポートする MIPS 命令

以下に、本アセンブラと tmips でサポートする命令を示す。
（一部、履修生が実習により tmips に追加する命令もある）

#### 3.9.1. R 形式命令

オペランドとして 3 つのレジスタを持ち、最初のレジスタがディスティネーションレジスタ、残りの二つがソースレジスタとなる。

- add
  加算
- sub
  減算
- and
  論理積
- or
  論理和
- nor
  論理和の否定
- slt
  比較。第 1 ソースレジスタが第 2 ソースレジスタより小さければディスティネーションレジスタに 1 を、そうでなければ 0 を代入する。

例：  add $t0, $t1 $t2   ($t0 <- $t1+$t2)

#### 3.9.2. I 形式命令

ソースオペランドの一つあるいは一部が即値（定数）となる命令。I 形式命令は、さらに算術・論理演算命令、分岐命

令、メモリアクセス命令に分類できる。

#### 3.9.2.1. 算術・論理演算命令

3 つのオペランドを持ち、ディスティネーションと第 1 ソースオペランドがレジスタ、第 2 ソースオペランドが即値となる。

- addi
  加算
- andi
  論理積
- ori
  論理和
- lui
  即値をディスティネーションレジスタの上位 16 ビットに代入する。第 1 ソースレジスタは 0 番レジスタとする。

```
例： addi $t0, $t1, 10
     lui $t1,$zero,hi(F001)
     ori $t1,$t1,lo(F001)
```

#### 3.9.2.2. 分岐命令

2 つのオペランドをレジスタとして持ち、これらの比較結果により第 3 オペランドの即値で示された命令に分岐する。第 3 オペランドは分岐命令からのオフセットに自動的に変換されエンコードされる。

- beq
  二つのレジスタの値が一致していれば分岐する

```
例： beq $t0,$t1,LABEL
```

#### 3.9.2.3. メモリアクセス命令（ロード命令・ストア命令）

第 1 オペランドはレジスタ、第 2 オペランドはメモリオペランドとなる。

- lw
  ロード命令。32 ビットの値をメモリオペランドで示されたアドレスからロードし、第 1 オペランドのレジスタに代入する。
- sw
  ストア命令。レジスタ中の 32 ビットの値をメモリオペランドで示されたアドレスにストアする。

```
例：
    lw $t0,0($t1)
    sw $t2, -4($sp)
```

### 3.9.3. J形式命令

オペランドとして即値を一つ持つ。この形式の命令はJ命令のみであり、即値で示されたアドレスにジャンプする。

例： j LABEL