

# 計算機アーキテクチャ チートシート

小野とか

2024 年 11 月 22 日

## 1 命令対応表

### 1.1 機械語

#### R 形式

表1.1: R 形式機械語の構成

OP	rs	rt	rd	sham	func
6bit	5bit	5bit	5bit	5bit	6bit

OP オペレーションコード（命令操作コード, オペコード）

rs 第 1 ソースのオペランドレジスタ

rt 第 2 ソースのオペランドレジスタ

rd ディスティネーションのオペランドレジスタ（結果が入る）

sham ビットシフト

func 機能、命令フィールドのバリエーションを示す（機能コード）

#### I 形式

表1.2: I 形式機械語の構成

OP	rs	rt	constant or address
6bit	5bit	5bit	16bit

OP オペレーションコード（命令操作コード, オペコード）

rs 第 1 ソースのオペランドレジスタ

rt 第 2 ソースのオペランドレジスタ（転送データが入る）

16bit 定数またはアドレス

定数の場合には, 15bit で絶対値を示し, 左の 1bit は符号を表す

## 1.2 MIPS レジスタ表

ポインタはアドレスのようなもの. `scanf` で `&` をつけたけど, あれでポインタになる.

表1.3: MIPS レジスタ表

名称	レジスタ番号	用途	スタックの有無
\$zero	0	定数値ゼロ	-
\$v0 - \$v1	2 - 3	結果と式の評価	無
\$a0 - \$a3	4 - 7	引数	有
\$t0 - \$t7	8 - 15	数値演算レジスタ	無
\$s0 - \$s7	16 - 23	アドレスレジスタ	有
\$t8 - \$t9	24 - 25	予備のレジスタ	無
\$gp	28	グローバルポインタ	有
\$sp	29	スタックポインタ	有
\$fp	30	フレームポインタ	有
\$ra	31	戻りアドレス	有

## 1.3 MIPS 命令表

次のページに載ってます.

表1.4: MIPS 命令表

命令	表記例	意味	形式	備考
add	add \$s1, \$s2, \$s3	\$s1=\$s2+\$s3	R 形式	加算をするよ！
subtract	sub \$s1, \$s2, \$s3	\$s1=\$s2-\$s3	R 形式	減算をするよ！
add im	addi \$s1, \$s2, 4	\$s1=\$s2+4	I 形式	定数との加減算
load word	lw \$s1, 4(\$s2)	\$s1=memory[\$s2+4]	I 形式	メモリからレジスタへ
store word	sw \$s1, 4(\$s2)	memory[\$s2+4]=\$s1	I 形式	レジスタからメモリへ
and	and \$s1, \$s2, \$s3	\$s1=\$s2&\$s3	R 形式	bit 単位の AND
or	or \$s1, \$s2, \$s3	\$s1=\$s2 \$s3	R 形式	bit 単位の OR
nor	nor \$s1, \$s2, \$s3	\$s1= (~\$s2)&\$s3	R 形式	bit 単位の NOR
and im	andi \$s1, \$s2, 100	\$s1=\$s2&100	I 形式	定数との bit 単位の AND
or im	ori \$s1, \$s2, 100	\$s1=\$s2 100	I 形式	定数の bit 単位の OR
shift left logical	sll \$s1, \$s2, 10	\$s1=\$s2<<10	I 形式	定数分左シフト
shift right logical	srl \$s1, \$s2, 10	\$s1=\$s2>>10	I 形式	定数分右シフト
branch on equal	beq \$s1, \$s2, Label	if(\$s1==\$s2)goto Label	I 形式	等しいとき分岐
branch on not equal	bne \$s1, \$s2, Label	if(\$s1!=\$s2)goto Label	I 形式	等しくないとき分岐
set on less than	slt \$s1, \$s2, \$s3	if(\$s2<\$s3)\$s1=1; else \$s1=0;	R 形式	より小さいか (beq+bne)
set on less than im	slti \$s1, \$s2, 100	if(\$s2<100)\$s1=1; else \$s1=0;	I 形式	定数値よりも小さいかの分岐
jump	j Label	goto Label	無所属	目的の label への無条件分岐
jal	jal Label	良い例が思いつかない	わからん	行き (アド레스を a0 に入れる)
jr	jr \$ra	上に同じく	わからん	帰り (\$ra を入れようね)

ここから先は持ち込み不可ゾーンです.

## 2 アセンブリ言語例

### 2.1 if 文

1: C 言語での if 文

```
1 if(i == j){
2     f = g + h;
3 } else {
4     f = g - h;
5 }
```

2: MIPS 言語での if 文

```
1 /*****
2 $s0=f, $s1=g, $s2=h, $s3=i, $s4=j
3 *****/
4
5     bne $s3, $s4, Else // falseのときExitへジャンプ
6     add $s0, $s1, $s2 // trueのときの動作
7     j   Exit          // 強制的にExitへ
8 Else: sub $s0, $s1, $s2 // elseの処理
9 Exit:
```

### 2.2 ループの組み方, for 文

3: C 言語での for 文

```
1 int A[4], X=0;
2
3 for(int i=0; i<4; i++){
4     X = X + A[i];
5 }
```

4: MIPS 言語での for 文

```
1 /*****
2 $s0=A's 1st address, $t0=i, $t1=X, $t2=4
3 *****/
4
5     add $t0, $zero, $zero // i=0とする
6     add $t1, $zero, $zero // X=0とする
7     addi $t2, $zero, 4    // ループの回数
8     add $s1, $zero, $s0   // 開始アドレスのコピー
9 Loop: beq $t0, $t2, Exit  // $t0=$t2なら脱出
10    lw  $t3, 0($s1)       // $t3に所定の場所のAをコピー
11    add $t1, $t1, $t3     // X=X+A[i]
12    addi $s1, $s1, 4      // アドレスのインクリメントに相当
13    addi $t0, $t0, 1      // 回数のインクリメント
```

### 2.3 ループの組み方, for 文

5: C 言語での for 文

```
1 int A[4], X=0;
2
3 for(int i=0; i<4; i++){
4     X = X + A[i];
5 }
```

6: MIPS 言語での for 文

```
1 /*****
2 $s0=A's 1st address, $t0=i, $t1=X, $t2=4
3 *****/
4
5     add $t0, $zero, $zero // i=0とする
6     add $t1, $zero, $zero // X=0とする
7     addi $t2, $zero, 4    // ループの回数
8     add $s1, $zero, $s0   // 開始アドレスのコピー
9 Loop: beq $t0, $t2, Exit  // $t0=$t2なら脱出
10    lw  $t3, 0($s1)       // $t3に所定の場所のAをコピー
11    add $t1, $t1, $t3     // X=X+A[i]
12    addi $s1, $s1, 4      // アドレスのインクリメントに相当
13    addi $t0, $t0, 1      // 回数のインクリメント
```