SCCP-021

A Peek Inside Computers

MIPS によるコンピュータ内部処理の理解

S1180180　Takahiro Sato

○目的
C 言語などで書かれたプログラムがどのような形で、コンピュータ内部で処理され実行されているのかを理解するため。そのために、高水準言語から低水準言語に落された際のアセンブリ言語に触れることで、内部の処理を具体的に理解していくことが出来ると考えた。

○MIPS
プロセッサには、大きく分けて RISC、CISC 方式がある。MIPS は RISC 方式。
RISC は、主な特徴として、固定命令語長であり動作が CISC に比べて高速である。
MIPS アーキテクチャは、任天堂やソニーなどで使われている。

―メモリ
0x00400000から下にテキストセグメント領域
0x10010000から下にデータセグメント領域
0x7FFFFFFF から上にスタックセグメント領域
0x00000000ー0x7FFFFFFF まで2^32通り表現出来る。よって、32bit ではメモリは4GB までの領域を扱う。

―CPU
CPU は中央処理装置であり、メモリからデータを受け取り、定められている命令セットに応じて処理する。処理した結果は、外部の装置に出力される。
主に、演算装置とレジスタで構成されている。

―レジスタ
幅32ビット、レジスタ数32個
$0
常に値が0であるレジスタ。
$1
la 命令をした際に示したラベルの、ベースアドレスが格納される。
$2
システムコールする際のサービスコードを入れる。また、入力された値を得る。
$4
アドレスを示したり、値を格納する。lw、sw、システムコールの際に、$4が示すアドレスや値が使われる。
$8-$15
一時レジスタであり、値やアドレスを保持する。

―システムコール
1
整数型(int 型)で$4に格納されている値を表示する。
4
$4で示しているアドレスの文字列を表示する。
5
キーボードから入力を得る、得た値は、$2に格納される。

ー使用した命令
R 形式

op	rs	rt	rd	shamt	funct

命令　レジスタ(rd), レジスタ(rs), レジスタ(rt)

・加算、減算
add　$8,　$9,　$10
$8 に $9 と $10 が加算された値が入る。
addu　$8,　$9,　$10
値を符合なしで加算を行う。
sub　$8,　$9,　$10
$8 に $9 から $10 を引いた値が入る。

・比較
slt　$2,　$9,　$10
$9<$10 ならば $2 に1が入る。それ以外なら0 が入る。

・論理演算
論理和(OR)
or　$8,　$8,　$9
$8 と$9 で論理和を取ったものを、$8 に入れる。
算術シフト(左シフト)
sll　$8,　$8,　2
$8 を2ビット左シフトし、$8に格納する。空いたビットは0で埋める。

・ジャンプレジスタ
jr　$ra
$ra に格納されているアドレスへジャンプする。ラベル main のアドレスを呼び出した時の、次のアドレスが格納されている。これは、元々いたアドレスに戻るためであるので、それを終了に使用した。

L 形式

op	rs	rt	address(即値)

命令　レジスタ(rt), レジスタ(rs), address(即値)

・加算、減算
addi　$8,　$9,　1
$9 と1(即値) を足した値を$8 に入れる。即値は、符合拡張された値。

・比較
slti　$2,　$8,　2
$8 < 2 ならば $2 に1が入る。それ以外なら0が入る。

・分岐
beq　$2,　$0,　exit
$2 が0 ならば ラベル exit にジャンプする。
bne　$2,　$0,　exit
$2 が0 以外ならば ラベル exit にジャンプする。
内部では、
beq　$2,　$0,　-24
bne　$2,　$0,　-24
現在いるアドレスから、相対的に値を取りジャンプしている。ここでは、現在いるアドレスから、24(0x18)戻ったところにジャンプする。

・ロード、ストア
lw　$8,　0($4)
$8 に $4 が示すアドレス内の値をロードする。
sw　$8,　0($4)
$4が示すアドレス内に$8の値をストアする。
la　$4,　input
ラベル input のアドレスを$4にロードする。
内部では、
lui　$1,　4097
ori　$4,　$1,　48
下位4ビット4097(0x1001)が $1 の上位4ビットにロードされ、ベースアドレスとなる。
ベースアドレス$1と48(0x30)で ori で論理和をとりアドレスを進め、ラベル input のアドレスを $4 にロードする。

ー MIPS 疑似命令
.asciiz　　“Word”
Word の最後に Null(0x00)文字が入りメモリに格納される。
MIPS は、ビッグエンディアンにより、文字の先頭からメモリに格納されていく。

d(0x64)	r(0x72)	o(0x6f)	W(0x57)

←――――― アドレス番地増加

.word　0x1234
値0x1234を1ワード使い、メモリに格納する。

00	00	12	34

○メモ
・ループ(if　n< 0)

```
input:  addi    $2,     $0,     5
        syscall
        slt     $4,     $2,     $0
        bne     $4,     $0,     input
```

・ストア(out に値をストア)

```
la      $4,     out
add     $4,     $4,     $9
sw      $8,     0($4)
```

○プログラム作成での注意点
例) 2 ～ 5 の数字を得る場合、エラー処理を設ける。

```
input:  addi    $2,     $0,     5
        syscall
        slti    $4,     $2,     2
        bne     $4,     $0,     input
        slti    $4,     $2,     6
        beq     $4,     $0,     input
```

範囲を指定することにより、後に続く動作でのエラー発生を防止する。

○まとめ、今後
アセンブリ言語では、C 言語では分からないレジスタやメモリといった、内部での動作を確認しながらプログラミングすることができ、少しではあるが理解できた。この内部の動作を知ることは、コンピュータそのものを知ることに繋がると思うので、まだ何に役立つか分かりませんが応用が効くものだと考えています。使用したレジスタや命令は少ないので、他のレジスタがどのような役割なのか、残りの多くの命令はどんな動作のためなのかを、理解していきたいです。また、ある程度 MIPS の知識が身に着き、理解した段階になったら、アセンブリならではでできることに取組んでみたいと思います。