手を動かして学ぶ！コンピュータアーキテクチャとアセンブリ言語プログラミングの基本

CPUとメモリのお話

この章は座学的な章になります。手を動かしてどんどん進めたい気持ちはわかりますが、基本的な用語や考え方を抑えずに進めてしまうと、思わぬところでつまずいてしまうものです。一回落ち着いて一通り見てみてください。全部一気に覚える必要はないですよ。まずは一般的な話から始めましょう。

bitとは

bitとはコンピュータの中の最小単位で、あるかないか、高いか低いかなど二つの状態を表すことが出来ます。それを使ってどのように数値を表現するのか。人によっては指折りで計算する時を思い出すかもしれませんね、あれは立てている指を1として、立っている指の本数で数を数えます、その方法だと両手を使ったとき単純に0から10まで数えることができます。確かにシンプルで人間にやさしい表現かもしれませんが、何か無駄が多い気がしますよね。

一回bitがどうとかは忘れて、表と裏があるコイン二枚について考えてみましょう。コインには大小がありコイン同士は見分けることが可能です。このコインたちを表裏を気にしつつ置いたとき、何通り考えられますかという問題です。表にすると以下のようになります。

No.	コインA	コインB
0	裏	裏
1	裏	表
2	表	裏
3	表	表

つまりコイン二枚がある時、四通りの表現ができるということです。この使えるもののなかで何種類表現出来るかというのがとても大切です。

話をbitに戻します。二つのコイン、つまり2bitあれば四種類状態を表現できることがわかりました。これを数値に置き換えるわけですが、じゃあbitが増えたときにどのように表現できる状態が増えるのか考えたときに、単純計算でn枚のコインがあったとき2のn乗だけ状態があるとわかりますね。また、普段使っている数、10進数と同様、それぞれの桁に重みを付ければ数値が表現できるのはわかりますか？じゃあ2bitを例にやってみましょう。

一桁目の重みを1、二桁目の重みを2としたとき、

二進数	対応する10進数
00	0x2 + 0x1 = 0
01	0x2 + 1x1 = 1
10	1x2 + 0x1 = 2
11	1x2 + 1x1 = 3

と対応するわけです。x桁目の重みを2のx-1乗としたとき、一般的な二進数の完成です。

それでは、負の数はどうやって表現しましょう、一番わかりやすいのが、一番左側のbitを+か-を表すものに割り当てるという方法です。確かに分かりやすいですが、+0と-0が区別できることにあまり利点はなく無駄になってしまいます。

そこで考えられたのが2の補数表現です。2の補数表現とはある数値のbitを反転して1を足したものをもとの数値の符号を反転した数として扱うという考え方です。この方法を使った場合のメリットは、引き算と足し算が論理的には同じになることや、先ほどのような先頭bitを正負の為に使うような方法でててしまう無駄が出ないかつ先頭bitを調べれば正負が判定することが出来ることにあります。4bitを例に説明します。

bit	ただの2進数	2の補数表現
0000	0	0
0001	1	1
0010	2	2
0011	3	3
0100	4	4
0101	5	5
0110	6	6
0111	7	7
1000	8	-8
1001	9	-7
1010	10	-6
1011	11	-5
1100	12	-4
1101	13	-3
1110	14	-2
1111	15	-1

例えば3に符号反転する場合を考えます。3のbitは0011です。まずbit反転をすると1100、そこに1を足すと1101、表の-3になっていますね。逆にマイナスからプラスにすることも可能です。-8を符号反転します。-8のbitは1000、反転すると0111、1を足すと1000、2の補数表現では表現不可能ですが、ただの二進数まで考えたときに、上手く反転できているのが分かると思います。

他にもbitで実数を扱う方法など数値表現はまだまだあるのですが、このMLFEには実数を扱う計算機が無いので省略することにします。

それでは、bitで数値を扱うことが出来るようになりました。数字が扱えるようになったら人間が扱いやすいように文字も使いたいですよね。これはさらっと紹介しましょう。ASCIIコードというものです。

10		10		10		10	
0	NULL	32	SP	64	@	96	`
1	SOH	33	!	65	A	97	a
2	STX	34	"	66	B	98	b
3	ETX	35	#	67	C	99	c
4	EOT	36	$	68	D	100	d
5	ENQ	37	%	69	E	101	e
6	ACK	38	&	70	F	102	f
7	BEL	39	`	71	G	103	g
8	BS	40	(	72	H	104	h
9	HT	41	)	73	I	105	i
10	LF	42	*	74	J	106	j
11	VT	43	+	75	K	107	k
12	FF	44	,	76	L	108	l
13	CR	45	-	77	M	109	m

10		10		10		10	
14	SO	46	.	78	N	110	n
15	SI	47	/	79	O	111	o
16	DLE	48	0	80	P	112	p
17	DC1	49	1	81	Q	113	q
18	DC2	50	2	82	R	114	r
19	DC3	51	3	83	S	115	s
20	DC4	52	4	84	T	116	t
21	NAK	53	5	85	U	117	u
22	SYN	54	6	86	V	118	v
23	ETB	55	7	87	W	119	w
24	CAN	56	8	88	X	120	x
25	EM	57	9	89	Y	121	y
26	SUB	58	:	90	Z	122	z
27	ESC	59	;	91	[	123	{
28	FS	60	<	92	\	124	|
29	GS	61	=	93	]	125	}
30	RS	62	>	94	^	126	\~
31	US	63	?	95	_	127	DEL

ASCIIコードとは、8bitでアルファベットとか記号とか改行とかを表すために考えられたものです。改行は10とか13であることとか、大文字に32足せば小文字になることとか、全てを暗記する必要はありませんが、ちょっと覚えておくと便利です。

これで、数値・文字とよく使うようなデータを定義することができました。これからは特に断りなく使うので、分からなくなったら遠慮せず何度も見返してみてください。

CPUとは

CPUとは、Central Processing Unitという日本語で言うなら中央演算処理装置のことです。中に色々な計算をするための回路とか数値を保持しておくための仕組みなどが詰まっており、導線を介してメモリに書き込んだり入出力を行ったりします。

とても基本的なCPUについて考えます。そのCPUには演算装置として算術論理演算装置（ALU）、汎用レジスタ（GR）、フラグレジスタ(FR)があります。また制御装置としてプログラムカウンタ（PC）、命令レジスタ（IR）、デコーダ（DEC）が実装されています。 それぞれについて表にまとめます。

名前（略称）	意味
算術論理演算装置（ALU）	四則演算とかbit演算とかを行う回路
汎用レジスタ（GR）	計算した結果を保持したりいろんなことに使うレジスタ
フラグレジスタ（FR）	計算結果を基に変化するレジスタ
プログラムカウンタ（PC）	次に実行する命令のアドレスが入っているレジスタ
命令レジスタ（IR）	実行する命令を保持するレジスタ
デコーダ（DEC）	命令レジスタから受け取りどんな処理をするか解読する回路

CPUの中では、『命令の取り出し』→『命令の解読』→『命令の実行』→『命令の取り出し』→...というサイクルで行われています。

『命令の取り出し』はプログラムカウンタを基にメモリから次の命令を取り出し命令レジスタに格納する。
『命令の解読』は命令レジスタに入っているものをデコーダに伝え、次に何をするかを理解する。

『命令の実行』はデコーダでわかった内容の通りにALUやGRを使い計算をする。計算結果次第でFRの値が変わる。

CPUはそういうものです。

それでは、CPUの種類について話しましょう、『8bitのマイコン』とか『64bitのCPU』などの言葉に耳覚えはありませんか？これはCPUのレジスタのサイズやメモリなどそのCPUの中でよく使われているbit長を特徴として取り上げて言われるものです。例えば、パソコンによく搭載されているCPUにIntelがありますが、その命令セットであるIA-32の仕様では、32bitの汎用レジスタが8個実装されており32bitのCPUであることがわかります。

またRISCとCISCという分類もあります。RISC(Reduced Instruction Set Computer)とは、命令を簡単なものに絞りハードウェアを簡単にすることで高速化をはかった考え方のCPUのことで、CISC(Complex Instruction Set Computer)とはそんなRISCが考えられる以前のCPUのことです。例えばスマートフォンによく搭載されるCPUにARMというものがありますが、ARMはRISCの考え方に則ったCPUです。下に比較のための表を置いておきます。

項目	CISC	RISC
命令数	多い	少ない
命令長	可変	固定
命令実行のクロック数	可変	固定
高速化手法の導入	困難	容易
ハードの開発期間	長い	短い

上記のような知識を前提として、CPUの特徴を端的に述べる際によく『32bitのRISCアーキテクチャ』というような言い方をします。意味はなんとなくわかりますよね？

メモリとは

メモリとは、データや命令をためておくための装置です。一般的なパソコンでは、『RAM』と呼ばれ8GBとか32GBとか言われているあそこです。でも先ほどCPUの説明の中でもデータをためておくことのできる装置がいたのですが覚えていますか？そう、レジスタです。しかしレジスタは32bit保存できるものが8本とかしか実装されていません。これはメモリと比べるとあまりにも貧弱に見えますが、きちんと役割があります。下の表を見てください。

名称	アクセス速度(秒)	記憶容量(バイト)
レジスタ	100p-10n	8-1K
キャッシュ	1n-20n	128K-8M
メモリ	10n-300n	256M-64G
HDDやSSDなど	1μ-10m	512M-12T

表を見てみると、アクセス速度が速いものは記憶容量が乏しく、記憶容量が多いものはアクセス速度が遅いです。このようにデータをためる部分にはトレードオフと言える関係があり、役割が違います。メモリ関係で覚えておくことは、記憶装置はトレードオフの関係があり、メモリアクセスが多ければ遅いソフトウェアになってしまうということです。

MLFEの仕様について

それでは、MLFEの仕様について説明致しましょう。MLFEはNABY(Newbew Architecture By Y)というCPUの上で動く命令セットです。Yとは製作者の頭文字です。MLFE（Machine Language For Education）とは、教育用途の機械言語という意味です。この二つの関係は、日本人と日本語のような関係で、動いているものとその上に乗る言語のようなものと思ってください。
MLFEは基本情報技術者のCASL-2を基に32bitRISCへ拡張したものです。ということはNABYはCOMET2を拡張したものということになりますね。
ここからは専門用語的な話になります、分かる人も分からない人も頭の隅に置いておくくらいで大丈夫。
NABYのレジスタは命令レジスタ、プログラムカウンタ、スタックポインタ、フラグレジスタ、後は14個の汎用レジスタがあります。表にまとめます。

名称(略称)	個数	bit	意味
命令レジスタ(IR)	1	32	実行する命令を保存するレジスタ
プログラムカウンタ(PC)	1	16	次に実行する命令の番地を保存するレジスタ

名称(略称)	個数	bit	意味
スタックポインタ(SP)	1	16	スタックの先頭を保持するレジスタ
フラグレジスタ(FR)	3	1	計算結果に応じて変化するレジスタ、正負・零か否か・オーバーフローの有無
汎用レジスタ(GR)	14	32	計算結果を保持したりいろんなことに使えるレジスタ

この中で特に汎用レジスタはよく使います。覚えておいてくださいね。

NABYはメモリの範囲として16bit使用することが出来ます。つまり0から65535番地まで使うことが出来るということです。メモリの１つは32bit保持することが出来ますので、256KiB使えるというわけですね。昨今ではそこら辺の画像ファイルよりも小さいですが、用途を絞れば結構いろんなことが出来ますよ。

計算できる種類としては、四則演算・論理演算・比較演算・シフト演算を行うことが出来ます。浮動小数点はサポートしておらず、標準では実数計算はできません。

今回はこのあたりで、おいおいしっかり紹介します。

まとめ

- bitとはコンピュータの最小単位、数値や文字の表現に用いることができる。
- CPUは中央演算処理装置のこと、レジスタとかALUとかが詰め込まれている。
- メモリはデータをためる装置のこと、速いと少なく多いと遅いトレードオフの関係がある。メモリアクセスはレジスタ同士よりも遅い。
- MLFEとはCASL-2を32bitRISCへ拡張したもの。
- MLFEが扱うメモリの範囲は16bit、256KiB使うことが出来る。
- MLFEは整数演算をサポートしている。