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アーキテクチャと入出力の話

今回はよりコンピュータアーキテクチャについて深く学びます。座学は今回で最後なのでがんばって！なお、MLFEとはあまり関係ない内容も知識や教養として紹介します。

ノイマンアーキテクチャとは

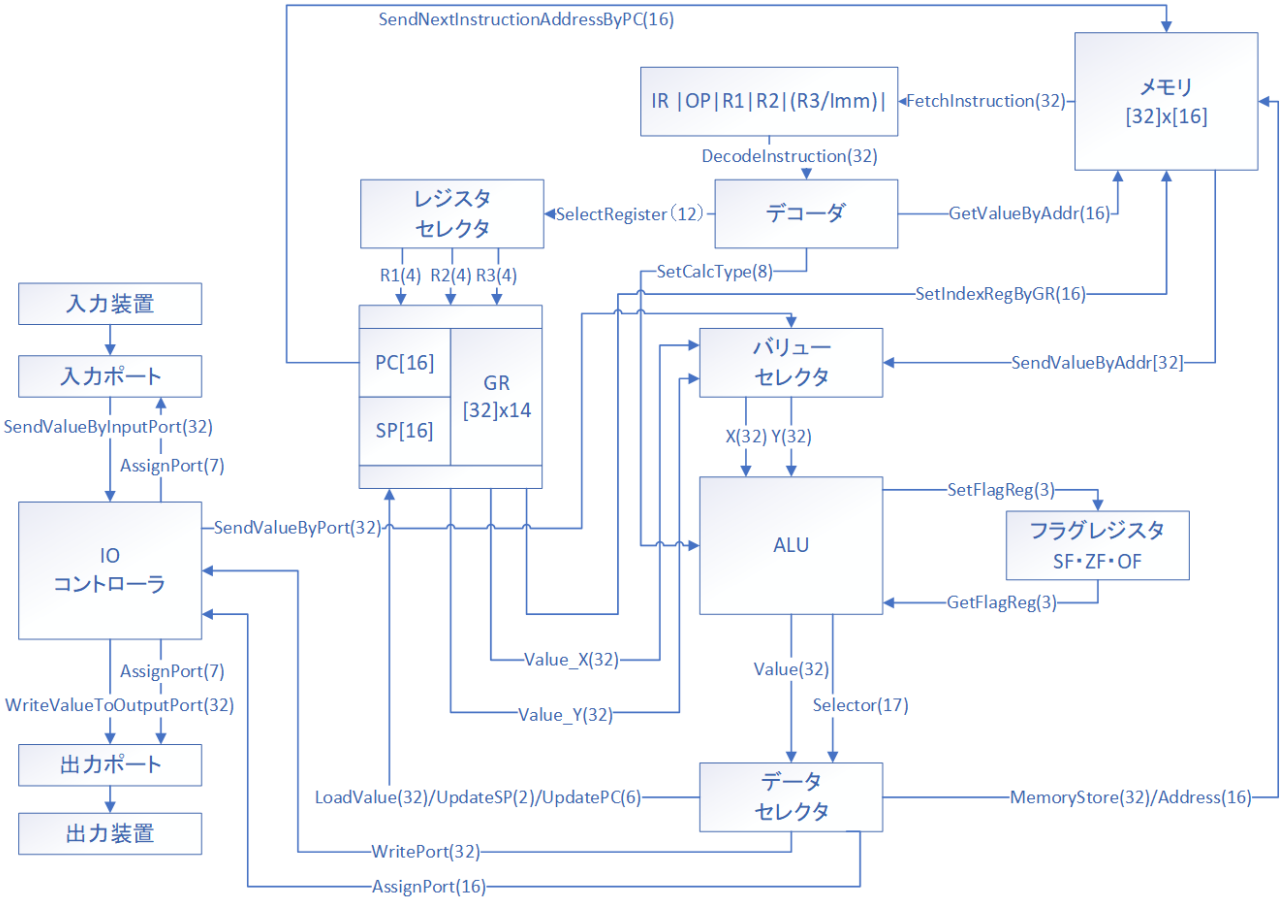
ノイマン型コンピュータとは、プログラム可変内蔵方式、逐次処理、単一メモリを特徴とする現在のコンピュータに多く採用されている方式です。

名称	意味
プログラム可変内蔵方式	プログラムを内部のメモリに記憶させることで、 プログラムを簡単に変更することができるということ。
逐次処理方式	命令は実行順にメモリに格納されており、順次取り出し処理すること。
単一メモリ方式	命令とデータは同じメモリに格納すること。

基本的な構成としては、演算装置、制御装置、記憶装置、入力装置、出力装置があります。

名称	意味
演算装置	算術演算や論理演算を行う装置。
制御装置	他の装置を制御する装置。演算装置と合わせてCPUに収まっている。
記憶装置	データやプログラムを記憶しておく装置。
入力装置	プログラムやデータを主記憶装置に入力する装置のこと。
出力装置	データを出力する装置のこと。

MLFEではこのノイマン型コンピュータを採用している。以下の図を基にどのように命令が実行されているか説明します。



NABYコンピュータモデル図

まずプログラムカウンタ(PC)からメモリに対して次の命令のアドレス番地を送ります。その命令は命令レジスタ(IR)に送られます。これをフェッチと言います。
次に、命令レジスタの内容を確認するためにデコーダに送ります。デコーダは、どんな計算をするか・どのレジスタを使うか・どのメモリにアクセスするかなどフェッチされたbit列の解釈を行います。これをデコードと言います。
次に、解釈した結果を基にい必要な値を算術論理演算装置(ALU)に送り、その結果をレジスタに書き込んだり、外部に出力したりします。これを実行と言います。

このノイマン型コンピュータとはまた別の方式のコンピュータがあります。それはハーバードアーキテクチャです。簡単に説明すると、命令とデータのメモリが分かれていることが特徴です。ノイマン型コンピュータにはフォン・ノイマンのボトルネックという、メモリアクセスが多すぎて遅くなってしまうという問題があり、それを回避して高速に動作できる反面、ハードウェア構成が複雑になってしまう問題があります。現代のコンピュータの多くはノイマンアーキテクチャを採用しているという話をしましたが、しかし命令用のキャッシュメモリを備えるCPUや個別にメモリをもつグラフィックプロセッサなどがあるのが現実で、このような構成をハーバードアーキテクチャの特徴を持っていると言えます。ノイマンアーキテクチャとハーバードアーキテクチャは両者の良いところが融合している状態になっているんですね。

ワイヤードロジックとマイクロプログラム

前に言った通り、コンピュータはフェッチ・デコード・実行を繰り返し動作していますが、デコーダから出力されるデコード情報はALUやレジスタに与える制御信号となります。その制御信号をどのようにして実行まで持っていくかという方法は大きく分けて二つあり、ワイヤードロジック制御方式とマイクロプログラム制御方式です。

名称	特徴
ワイヤードロジック制御方式	デコード情報を配線によって直接送り込む。
マイクロプログラム制御方式	デコード情報を基にマイクロプログラムで処理する。

ワイヤードロジック制御方式では、命令が複雑になればなるほど配線が多くなり設計が難しくなるものの、RISCで設計されたCPUは命令が単純なものの集まりなのでワイヤードロジック制御方式で問題ない場合が多いです。NABYはワイヤードロジック制御方式を採用しています。

マイクロプログラム制御方式は複雑な命令をマクロ命令として、制御メモリと呼ばれる場所にあるマイクロ命令群を呼び出して実行します。CISCに採用される場合が多いです。一応間違えないで欲しいのは、マイクロプログラム制御方式でのマクロ命令

とMLFEにおいてのマクロ命令では意味が違うことに注意してください。

入出力

コンピュータには、キーボードやカメラなどの入力装置、ディスプレイやスピーカーなどの出力装置がありますが、それら入出力操作をコンピュータの深いところまで突き詰めていくと、方法としては二種類あることがわかります。それはメモリマップトI/OとI/Oマップト I/Oです。

名称	意味	例
メモリマップトI/O	特定のメモリ番地に入出力装置を割り当てる。	アドレスのX番地からX+7までマウスの クリックやホイールの情報が書き込まれる。
I/Oマップト I/O	入出力命令を用いてデータの送受信を行う。	WRITE命令を用いて レジスタの内容をポート0に出力する。

MLFEではI/Oマップト I/Oを採用しており、特定のポートに標準入力や標準出力が割り当てられています。ポートは入力ポートと出力ポートの二系統あります。

ポート番号	シミュレートデバイス
入力ポート0	1文字の標準入力
出力ポート0	1文字の標準出力
出力ポート1	符号付き10進数標準出力
出力ポート2	16進数標準出力
出力ポート3	2進数標準出力
出力ポート4	符号なし10進数標準出力

他にもあります、マニュアルを要参照。

例えば、レジスタGR1に標準入力の一文字を書き込む場合は

```
PROCA   LD      GR0, STDIN
        READ    GR0, GR1
STDIN   DC      0
```

また、GR2の値を符号付き10進数標準出力として出力する場合は

```
PROCB   LD      GR0, STDOD
        WRITE   GR0, GR2
STDOD   DC      1
```

メモリアドレッシング

メモリアドレッシングとは、ある特定のメモリ番地を指し示すための方法です。6つに分類されます。

名称	意味	MLFE対応
直接アドレッシング	命令オペランドに記述したアドレスのデータを対象にする。	o
間接アドレッシング	対象のアドレスの値を有効アドレスとしてデータにアクセスする。	x
指標アドレッシング	指標レジスタと即値を加算したものを有効アドレスとしてアクセスする。	o
相対アドレッシング	プログラムカウンタと即値を加算したものを有効アドレスとしてアクセスする。	o
基底アドレッシング	OSが管理するプログラムの先頭番地が入った基底レジスタと 即値を加算したものを有効アドレスとしてアクセスする。	x
即値アドレッシング	オペランドに記述した値をそのまま対象データとして扱う。	o

- 直接アドレッシングの例

```
PROCC   LD      GR1, VALUE
VALUE   DC      10
```

レジスタGR1にアドレスVALUEの値を格納。

```
GR1 <- Memory[VALUE]
```

- 指標アドレッシングの例

```
PROCD   LD      GR2, ARYSUB
        LD      GR1, ARRAY, GR2
ARYSUB  DC      1
ARRAY   DS      10
```

レジスタGR2に配列添え字が格納されるアドレスARYSUBの値を格納。
レジスタGR1にアドレスARRAY+GR2の値を加算した値を有効アドレスとしてその値を格納。

```
GR1 <- Memory[ARRAY + 1(GR2)]
```

- 相対アドレッシングの例

```
PROCE   LD      GR1, VALUE, PC
VALUE   DS      10
```

やっていることは、指標アドレスにプログラムカウンタを指定しているだけ。

- 即値アドレッシングの例

```
PROCF   LAD     GR1, 10
```

レジスタGR1に即値である10を直接格納。

```
GR1 <- 10
```

まとめ

- ノイマンアーキテクチャは `プログラム可変内蔵` `逐次処理` `単一メモリ` の特徴を持つ広く使われているコンピュータの形式である。
- ノイマンアーキテクチャの他にハーバードアーキテクチャというものがあり、例えば単一メモリでは無いなど特徴がある。
- 実行制御方式として `ワイヤードロジック制御` と `マイクロプログラム制御` がある。
- 入出力には `メモリマップトI/O` と `I/Oマップト I/O` がある。
- メモリアドレッシングは `直接` `間接` `指標` `相対` `基底` `即値` がある。
- MLFE(NABY)はノイマンアーキテクチャ、ワイヤードロジック制御、I/OマップトI/Oを採用している。
- MLFEは `直接` `指標` `相対` `即値` アドレッシングに対応している。