電子計算機 **3E**

3E　2004.2.5

本日の授業のテーマ

(1) コンピューターの動作原理

1 メモリーと CPU の関係

ここでは、コンピューターのメモリーと CPU の関係をシミュレーターWCASL II を通して学習する。コンピューターの最小構成要素は、CPU とメモリー(主記憶装置)である。コメットの場合、図１のようになっている。

本日は、このコンピューターの動作を COMET II のシミュレーターWCASL-II を通して学習する。

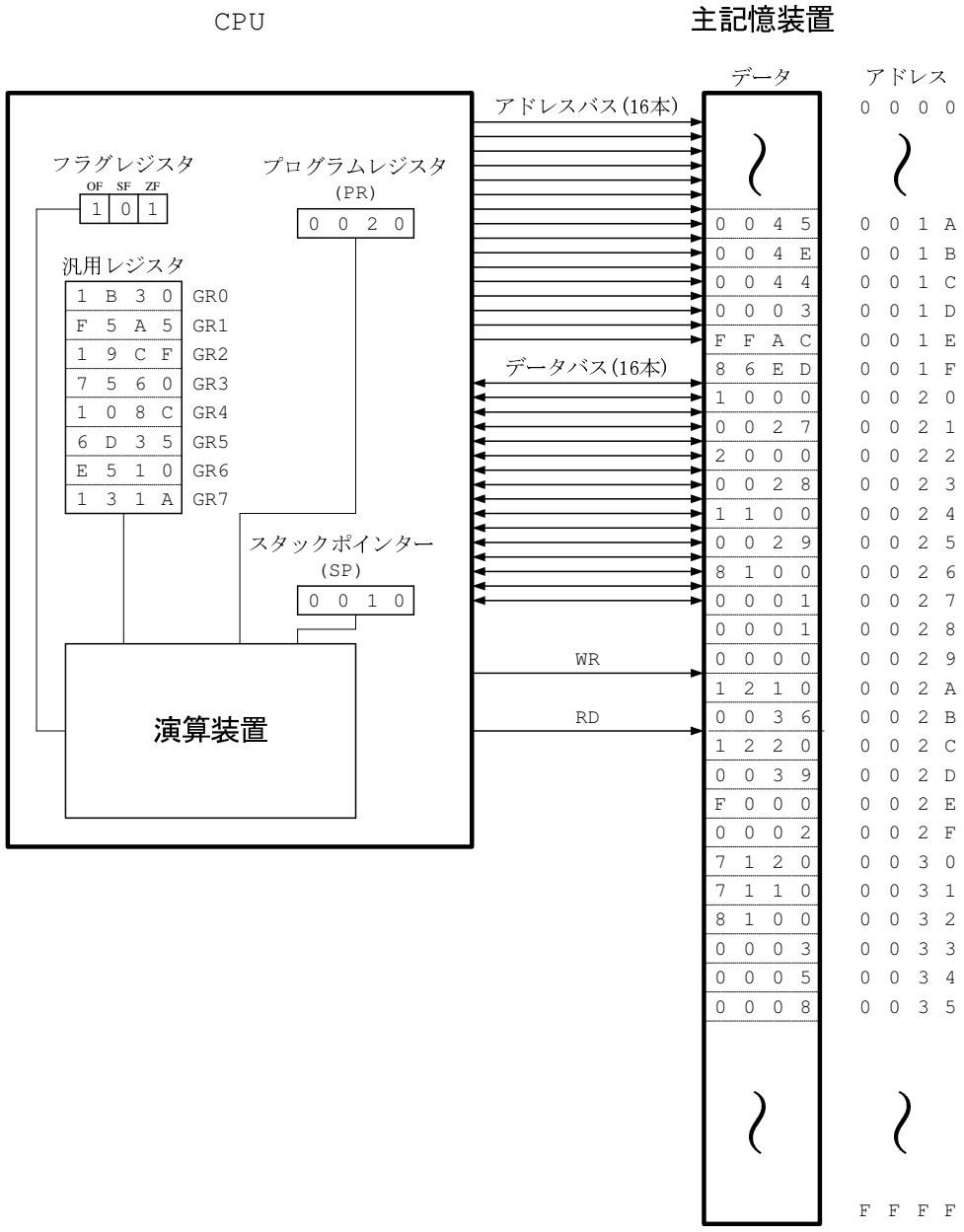


図１ COMET のハードウェアー

2 プログラムの実行例

ここでは、同じ動作をする 3 つのプログラムを実行して、コンピューターの動作を考える。2.1〜2.3 のいずれのプログラムも 1+1 の加算の計算を行い、メモリー上にその結果を格納するプログラムである。

2.1 普通の CASL II のプログラム

これは最も普通の CASL II のプログラムである。説明するまでも無く、ラベル A と B の値を加算して、ラベル C に格納数プログラムである。

```
PGM   START
      LD        GR0,A
      ADDA      GR0,B
      ST        GR0,C
      RET
A     DC        1
B     DC        1
C     DC        0
      END
```

[確認事項]

このプログラムを打ち込んで、実行させてみよう。そして、以下を確認しなさい。

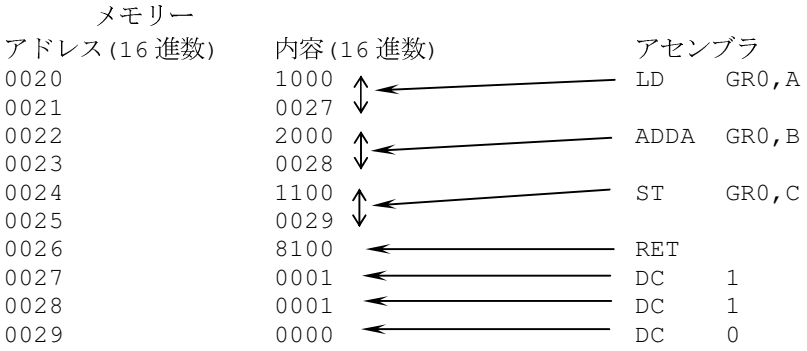
- アセンブルし、機械語のビットパターンを良く見よ。ビットパターンとアセンブラの関係が 1 対 1 になっていることを確認せよ。
- アセンブラ命令の DC は、その値がメモリーに格納されているだけである。これを確認せよ。
- 実際にプログラムを実行させて、機械語命令の実行されることを確認せよ。

2.2 機械語のプログラム

2.1 はアセンブラ言語で書いたものであるが、以下のプログラムは機械語のプログラムである。

```
PGM   START    NEXT
NEXT  DC       #1000
      DC       #0027
      DC       #2000
      DC       #0028
      DC       #1100
      DC       #0029
      DC       #8100
      DC       #0001
      DC       #0001
      DC       #0000
      END
```

これが機械語そのものであることは、2.1 のアセンブラ言語のプログラムをアセンブラーで機械語に変換し、メモリーに格納する様子を見るとすぐに分かる。ただし、実行開始アドレス NEXT は、#0020 としている。シミュレーターWCASL-II の場合、#0020 がデフォルトの実行開始番地である。アセンブラ言語と機械語の関係は以下のようになる。



機械語にアセンブルしてしまうと、もはやデータ(ラベル A, B, C)と命令の区別はなくなってしまう。アセンブラ言語の場合、まだデータと命令は区別できた。しかし、機械語になると全く区別できない。例えば、RET を表す機械語 8100 は、命令なのかデータなのかの分からない。当然、CPU もデータと命令の区別はしていない。

ではいったい CPU は間違えずに命令を実行するのか?。CPU が命令と考えるのは、プログラムレジスタが示すアドレスの内容だけである。プログラムレジスタが示すものは、命令と考えて、それを CPU に取り込んで、そのビットパターンに従い、動作しているだけである。たったこれだけの仕組みで、コンピューターは同じメモリ上にあるデータと命令を間違えずに実行しているのである。

このように、同じメモリー上に命令とデータがあるようなものをノイマン型コンピューターと言う。世界中のほとんどのコンピューターがこのノイマン型のコンピューターである。

もう一度、ノイマン型コンピューターの特徴をまとめると、

- 1 次元的に並んだメモリーがあり、そこにプログラム(命令)もデータも格納される。メモリーの内容は、自然数の番地で参照できる。

- メモリーに格納されたプログラム(命令)とデータの見かけ上の区別はない。プログラムをデータとして見ることも、データをプログラムとしてみることもできる。

[確認事項]

COMET II がノイマン型のコンピューターであることを確認しなさい。

2.3 プログラムを作りながら実行するプログラム

命令とデータが同じメモリ上にかかれているということは、プログラムによりプログラムを作ることができる。このプログラムは、命令が書かれている領域に、データを書き込んでいる。そして、書き込んだ領域に JUMP 文で飛んでいる。そして書かれたデータは、命令として解釈されている。このようにノイマン型コンピューターでは、自分でプログラムを書きながら実行するプログラムを書くことができる。自己増殖しているようなものである。これを上手に利用すれば、なかなか面白そうである。

```
PGM   START
      LAD     GR1,-1
ONE   NOP
TWO   NOP
      LAD     GR1,1,GR1
      LD      GR2,P,GR1
      ST      GR2,ONE
      LAD     GR1,1,GR1
      LD      GR2,P,GR1
      ST      GR2,TWO
      JUMP    ONE
A     DC      1
B     DC      1
C     DC      0
P     DC      #1000
      DC      #0032
      DC      #2000
      DC      #0033
      DC      #1100
      DC      #0034
      DC      #8100
      END
```

[確認事項]

データが命令になっていることを確認しなさい。