Linux/Alpha 活用講座

清水尚彦　nshimizu@keyaki.cc.u-tokai.ac.jp

Linuxによる8ビットCP/Mマシンの開発

暖かくなってきたせいか、ここのところ故障するマシンが多くなってます。DP264が不定期にハングアップするようになったかと思えば、GatewayのAthlonマシンが突然電源断になりました。結局どちらも入院させるはめになってしまいましたが、一足先に退院してきたAthlonマシンは、ハードディスクがフォーマットされただけ(!)で、何の対策もされずに退院させられてきました。仕方なく、学生が独自に調査してみたら「電源ユニットのファンが止まっていた」ということでした。やっぱり、連続稼働が長いマシンはサーバにしておかないと駄目なのかな？

PS2 Linuxを入手！

PS2 Linuxが限定ながら発売されて、初日は用意した2000台が8分で売り切れるというフィーバー振りだったようですね。私も申し込みをして、幸い2000台の中の1台を確保できました。もっとも、その後7000台の追加生産体制を整えるらしいですから、欲しい人にはだいたい行き渡ることになりますね。このマシンは、メインエンジンはMIPSアーキテクチャで、性能的にも大したことはなさそうなので、後はいかにベクトルユニット(VU)をアセンブラで駆動できるかが、高性能アプリケーションのための決め手になります。VUを用いた数値計算ライブラリが整えば、誰でも簡単に使えるようになります。配布台数の多さを考えても、誰かがライブラリ作成に着手すると見てもよいと思えますから、実用的に使いたいという人は、それから考えてもいいのではないでしょうか？

そもそもPS2そのものは、メモリの少なさから数値計算には無理があるので、β版の出荷は、コミュニティのソフト開発を当てにしているってところがあるんじゃないかと思っています(どうなんでしょうかね、本当のところ)。まともに使いたい人は、メモリもずっと多く必要になるでしょうから、コミュニティの成果が十分に発揮されれば、時期を狙って、数値計算サーバとしてエモーションエンジン搭載のワークステーションを、SCEが出すかもしれません。

なんて考えていると、あまり技術情報を公開しないこのマシンに頑張ってライブラリを整えるという気持が薄れてしまいますね。私はあまり手を付けないと思いますが、皆さん頑張ってください(って無責任モード)。取りあえずは、物が来るまでに「Sync on green」に対応するモニタを探しておかなくてはいけないなってのが当面の課題です。

Linuxによる8ビットCP/Mマシンの開発

今回は、ちょっと趣向を変えて「Linuxによる8ビットCP/Mマシンの開発」をお送りします。

5月号でCP/M(Control Program for Microcomputer)の話を少ししましたが、5月に行われたパルテノン研究会(記事末のRESOURCE[1]参照)で、本研究室からCP/Mマシンをフルスクラッチで作成するプロジェクトを発表しました。Linuxを用いた論理設計も、以前は最後のFPGA(Field Programmable Gate Alley)／CPLD(Complex Programmable Logic Device)ベンダツールのところでWindowsマシンが必要になっていたのですが、最近はどこのベンダも揃ってLinuxへのコミットを始めているので、もう少ししたらLinuxだけですべての開発ができるようになりますね。私が使っているALTERA社CPLDでは、Linuxをサポートするというアナウンスがなされています([2])。今のところ研究室では、取りあえずWindows版を使っ

表1　研究室で作成した実験用CP/Mマシンのシステム構成

CPU	命令セット	i8080A 互換命令＋拡張メモリアクセス命令＋シリアルインターフェイス拡張
	バス方式	MC6800 類似同期バス（PARTHENONで非同期バスは難しいので）
	実行制御	ハードワイヤ／ノンパイプライン（実はMCS6502程度のパイプラインを入れた論理ファイルもあるのですが……）
入出力	ROMディスク	1Mbit ROM
	RAMディスク	1Mbit RAM（半分はCPUのアドレス空間用）
	シリアルコンソール	今やどのPCでも簡単なターミナルプログラムを使って端末にできるから、ずいぶん簡単に済むものです
アプリケーション	ROMディスク中にCP/Mシステム	
	MBASIC（デモ用）	

ていますが……。

レトロなシステムが好きな人もいるかもしれないので（そりゃお前だって！）、今月号では、我々のやってきたCP/Mシステム開発の紹介をしましょう*1。

システム構成は表1に示す通りです。作成には、パルテック社のブレッドボードにメモリボードを外付けする形を採りました。CPLDの合成用のダウンロードケーブルも弱小研究室では買えないので自作しました（写真1）。

MBASICはマイクロソフト社のライセンス製品ですが、「Microsoft BASIC Version Information」（[3]）によると、最近のVisual Basicのライセンスを持つ人なら、マイクロソフトの古いBASIC製品を使用できることになるそうなので、デモ用に乗せてみました（マイクロソフトへの確認はしてません）。

CPUとUART（Universal Asynchronous Receiver Transmitter：シリアル用チップ）は、ALTERA社の「CPLD FLEX EPF10K30」か「FLEX EPF6016」に収まりますが、研究室にあるブレッドボードの「EPF10K50S」をターゲットとして合成します。CPUの設計は、命令セット表を入手して、その通りに作ればよいので割愛します。ここでは、CP/Mの移植と論理シミュレーションの話をします。

CPUのクロック周波数は、研究室の在庫品のROM/RAMのアクセスタイムが150ns程度というものなので、同期バスの半分の時間で十分な余裕を持ったアクセスが可能な2MHzを目標値としています。もちろん、まじめに作ればもっと速いクロックスピードでも動くし、MAX＋PLUS2の遅延評価では、10MHz近いクロック周波数で動作すると報告されています。ですが、ここはあくまでも在庫LSIの性能中心に考えています（当然ですが、この低い目標性能を満足するために回路の高速化の工夫はイッサイやっていません）。

まず、CP/Mを動かすときに考えることが、ディスクの問題です。その昔、CP/Mシステムを作ろうとして一番手間がかかったのがこの点です。ディスクコントローラを使ってフロッピーディスクをアクセスするためには、それなりの手間

写真1　開発中のシステムの様子
（左からCPLDダウンロード基板、ブレッドボード、RAM／ROM基板）

がかかるものなのです。それを小さなBIOSルーチンの中に収めてシステムを構築するためには、まずCP/Mが動くマシンが必要となりました。これが一番のネックで、アマチュアがCP/Mのパッケージを買ってきて自分で作ろうとしても、動くCP/Mマシンがないことには話が始まらないのです。

この点、今ではLinuxでZ80エミュレータが動作し、その上でCP/Mを動作させることができる訳ですから、「はじめの一歩」の敷居はかなり低くなりました。CPUの論理シミュレーションで命令動作の確認をしている間に、エミュレータでBIOSの開発とCP/Mの移植を行います。

我々が使用したエミュレータは「Yaze-1.10」（Yet Another Z80 Emulator）というものです。これを使った積極的な理由はあまりありませんが、結果としてROMディスクの開発などで、作業量が節約できて助かりました。さらに、ROMやRAMも大容量のものが使えるようになってきているので、昔の1Sフロッピー（片面単密フロッピー）などより、むしろ大きなROMが平気で使えるようになっています。そこで、実験用のシステムはROM／RAMディスクベースとすることにして、BIOSの開発の手間を大幅に縮小することにします。余裕があれば、ATAPIなどのインターフェイスを作るなどして、他のマシンとの連係を

*1　Alphaと何の関係があるんだと思う人もいるかもしれませんが、エミュレータのYazeをAlpha Linuxで動かしているから、かすめてはいるんです。論理合成／シミュレータのPARTHENONはx86版Linuxなのですが……:-）。

リスト1　cdm.cの変更箇所

```
●229行目
caddr_t cp;
      ↓
__ptr_t cp;

●243行目
if ((st.st_mode & S_IFMT) != S_IFREG) {
      ↓
if (!S_ISREG (st.st_mode)) {

●282行目
if ((cp = mmap(NULL, dp->size, prot, MAP_SHARED, dp->fd, 0)) == (caddr_t)-1) {
      ↓
if ((cp = mmap(NULL, dp->size, prot, MAP_SHARED, dp->fd, 0)) == (__ptr_t)-1) {

●1221行目
return (st.st_mode & S_IFMT) == S_IFDIR ? UX_DIR : UX_RFN;
      ↓
return S_ISDIR (st.st_mode) ? UX_DIR : UX_RFN;
```

向上させることもできるのですが、それはきちんと動いてから後で検討することにしました(CPLDでCPUを作っているので、簡単に論理も変更できるし、BIOSが書きやすいように命令を追加することもできます)。

ROMディスクやRAMディスクを汎用の8ビットCPUに接続しようとすると、バンク切り替えだの何だのと、いろいろ面倒なことをすることが多くなるようです。しかし、我々は専用CPUを作成するので、面倒なことをする必要はありません。ROMディスクやRAMディスクのためのアドレスを素直にCPUに拡張し、これらの拡張アドレスをアクセスする専用命令を2つ追加しました。

この追加命令のコードは、Z80とも重ならないようにして、将来CPUをZ80互換に拡張しようとしときにも困らないようにしておきます*2。これらの命令を使うことによって、BIOSの開発は極めて簡単になり、セクタリードのルーチンは、ただのメモリコピーに置き換わります。当然ながら、セクタスキューなんてものは使わないし、コンソール入出力も当面は割込みなしのハードウェアハンドシェイクにしておきます。ハードウェアハンドシェークが後でトラブルの原因になったのですが……。

エミュレータ「Yaze」を入手する

では、具体的な開発手順を示しましょう*3。

まずは、YazeのWebページ([4])に示すFTPサイトからエミュレータYaze-1.10を入手します。このソースファイルはPOSIXに準拠していないため、Linuxでコンパイルするには、ファイルcdm.c(**リスト1**)と、ファイルmonitor.c(**リスト2**)に修正が必要になります。修正が済んだら、次のコマンドでエミュレータのコンパイルとインストールを行います。

```
% make
% make install
```

エミュレータの配布ディレクトリ「YAZE-1.10」には、「test」

リスト2　monitor.cの変更箇所

```
●33行目(1行追加)
#include <time.h>

●239行目
if (((st.st_mode & S_IFMT) != S_IFREG) || st.st_size == 0) {
      ↓
if ((!S_ISREG(st.st_mode)) || st.st_size == 0) {

●402行目
if ((st.st_mode & S_IFMT) == S_IFDIR)
      ↓
if (S_ISDIR(st.st_mode))

●405行目
if ((st.st_mode & S_IFMT) != S_IFREG) {
      ↓
if (!S_ISREG(st.st_mode)) {

●907行目
long tickspersec = CLK_TCK;
      ↓
long tickspersec = CLOCKS_PER_SEC;
```

*2　Z80の未定義命令を用いるプログラムとは互換性がないのですが、それは勘弁してもらいましょう。
*3　さらっと書いていますが、研究室の大学院生の近君と一緒にかなり試行錯誤した結果のお話です。

というディレクトリがあります。YAZE-1.10ディレクトリでYazeを起動すると、このtestディレクトリをAドライブにマウントしてCP/M互換のOSが起動します(実行例1)。これは、YAZE-1.10ディレクトリにある「.yazerc」が、ディレクトリのマウントとエミュレータの起動を自動で行っているためです。ここで重要になってくるのがtestディレクトリにある「SYS.COM」ファイル(SYSコマンド)です。このコマンドを実行すると、CP/Mを終了してYazeのプロンプト「$>」に戻ることができます。

ここでは今特にやるべきことはないので、「SYS」と入力してCP/Mを抜けますが、いろいろ試してみたい方は、DIRなどのビルトインコマンドを実行してみると、雰囲気がつかめると思います。なお、Yazeを終了するには、Yazeのプロンプトで「quit」と入力してください。

実行例1　Yazeを起動する

```
% yaze

Yet Another Z80 Emulator Ver. 1.10, Copyright 1995,1998 Frank D.Cringle.
yaze comes with ABSOLUTELY NO WARRANTY; for details
see the file "COPYING" in the distribution directory.

Running 'yaze.boot'A>
```

実行例2　~/cpmディレクトリに必要なファイルを用意してyazeを実行する

```
% yaze
$> mount a ~/cpm
$> go

Yet Another Z80 Emulator Ver. 1.10, Copyright 1995,1998 Frank D.Cringle.
yaze comes with ABSOLUTELY NO WARRANTY; for details
see the file "COPYING" in the distribution directory.

Running '/usr/local/lib/yaze.boot'
```

実行例3　mbasicを実行する

```
A>mbasic
BASIC-80 Rev. 5.21
[CP/M Version]
Copyright 1977-1981 (C) by Microsoft
Created: 28-Jul-81
34872 Bytes free
```

CP/Mの配布ファイルを入手する

次にCP/Mの配布ファイルを用意します。入手先は「The unofficial CP/M web site.」ですが、5月号に記載したURLから移転しています([5])。ここからは、CP/Mを含むDRI(Digital Research社)のマニュアル、バイナリ、ソースコードが入手できます。同じサイトにDRIのソフトをまとめたアーカイブDRIPAK.ZIP(DRIPAK)も置いてあります([6])。こちらを利用したほうが、後で利用するものが全部入っていて便利です。

このDRIPAKの中から「CPM_2-2/SYSTEM」ディレクトリ以下のファイルをすべて適当なディレクトリに取り出します。これはCP/M Ver.2.2の正式な配布ファイルです。そのディレクトリには、先ほどのYAZE-1.10ディレクトリにあるtestディレクトリの中にあるsys.comも併せて入れておきます。例えば、このディレクトリを「~/cpm」としてYazeを起動すると、実行例2のようになります。今度は、トランジェントコマンドの多くがディレクトリに入っているので、STATコマンドなどが使えるようになっています。

ちょっと注意しなくてはいけないのは、Yazeでは、UNIXのディレクトリをマウントしたディスクは読み出し専用になっていて、書き込みができないことです。書き込みをするためには、新たにCP/Mイメージディスクを作る必要があります。これを行うツールが先ほどパッチを当てた「cdm」です。詳しくはドキュメントを参照していただくことにして、ここでは「createというコマンドを利用する」とだけ言っておきます。

Aドライブにあるコマンドは、MS-DOSで言うところの「パスが通っている」ことになるので、システム関係のコマンドはすべてUNIXのディレクトリに入れて、データファイルをイメージディスクに入れる使い方にしたほうが良いでしょう*4。また、Visual Basicのライセンスをお持ちの方は、先ほどDRIPAKをダウンロードしたサイト([6])中にMBASICのファイル「MBASIC.COM」もあるので、これをダウンロードして実行してみましょう。実行例3のように、懐かしいメッセージが出てきます*5。

SMALL CやFORTHなどCP/Mのフリーソフトだけでも結構遊べますし、CP/Mの正式配布ファイルを使わずにフリーのソフトだけで開発すれば、ライセンスにとらわれない形でかなりのことができます。

*4　これはオリジナルのCP/Mの機能ではなくて、YazeのZCPR(後述)の機能です。
*5　初めて使う方のために：MBASICからCP/Mに抜けるにはsystemコマンドを入力します。

64K CP/Mイメージを作成する

Yazeに使われているOSは先ほど「CP/M互換」と書きました。内部では、実際には「ZCPR」と「SUPERBDOS」というフリーのCCP／BDOS*6の互換ソフトが使用されています。

BIOSの呼び出しがあると、Yazeはこれをトラップして内部でエミュレーションするので、BIOSルーチンはエミュレーションされません。さて、フリーのものがあるのに、わざわざライセンスの制約のあるオリジナルを使うには訳があります。というのも、我々が作成したCPUはi8080A互換なのですが、フリーのソフトはZ80命令セットが必要となるのです。CPUのコアを小さくまとめるには、Z80よりも命令セットの小さなi8080Aの方がずっと適しているし、CP/Mは、そもそもi8080Aで動作するように作られているため、動作上も問題はないはずなのです。そこで、Yazeが用意しているフリーのソフトを使わずに、我々はDRI版の正式なCP/Mを用いることにしました*7。

といっても、Yaze自身はOSを持っている訳ではないので、ブートファイルを切り替えるだけなのです。ブートファイルとして利用可能なものには、CP/Mのシステム部分を取り出したファイルだけでなく、MOVCPMコマンドで作られたオフセットのかかったCP/Mシステムファイルも使えるようになっています。

実は、これがすごく役に立つのです。というのも、先ほどのDRIPAKの中のSYSTEMディレクトリには「CPM56.COM」というMOVCPMで作成されたCP/M OSイメージがあります。もちろんこちらはオフセットのかかったファイルになっています。Yazeの互換OSではMOVCPMは動作しないので、自分のCP/Mイメージを作成するには、まずはDRI版のCP/MでYazeをブートしなくてはなりません。

先ほどのようにファイルを展開していれば、

```
% yaze -b ~/cpm/CPM56.COM -p c4
```

のようにしてYazeを起動できます。オプションの「-p c4」は、エントリポイントのページを表します。指定するオプションが分からなければ、オプションなしでYazeを起動してみましょう。実行例4のように、必要なオプションの値が表示されるので、そのオプションを素直に付けてYazeを再起動してください。

実行例4 「-p c4」オプションなしで実行する

```
% yaze -b cpm/CPM56.COM

Yet Another Z80 Emulator Ver. 1.10, Copyright 1995,1998 Frank D.Cringle.
yaze comes with ABSOLUTELY NO WARRANTY; for details
see the file "COPYING" in the distribution directory.

Warning: cpm/CPM56.COM appears to need -p c4
Running 'cpm/CPM56.COM'
$>quit
```

実行例5 CP/Mファイルを64Kバイト用にリロケートする

```
A>movcpm 64 *

CONSTRUCTING 64k CP/M vers 2.2
READY FOR "SYSGEN" OR
"SAVE 34 CPM64.COM"
```

ところでこの「CPM56」というファイルの名前、気になりませんか？ これは「56K CP/M」といって、56Kバイトのメモリ専用のCP/Mファイルを表しています。我々は、i8080Aのメモリ空間64Kバイトをフルに使えるCP/Mが欲しいので、これを64K用にリロケートしなくてはなりません。オリジナルCP/Mでは、リロケート用にMOVCPM.COMというファイルが用意されていました。このコマンドは、システム空間のメモリイメージとコマンド内のリロケーションテーブルとの同期が取れていないと動作しません。そのため、ZCPRとSUPERDOSの組み合わせのYazeの配布ファイルでは動作しません。オリジナルのCP/MファイルであるCPM56.COMで起動されれば問題なく動作します（実行例5）。

このファイルは、デバッガDDTでロードすると、900H番地からブートローダがあり、本来E400Hから始まるCCPが980Hから始まるようにメモリ上に配置されているのが分かります。

```
0900H-097FH     ……ブートローダ
0980H-117FH     ……CCP
1180H-1F7FH     ……BDOS
1F80H-227FH     ……BIOS
```

自分のシステム用のBIOSを作ったら、これをアセンブルしてHEXファイルに変換し、CP/Mのイメージ上に上書き保存します。これで、新しいCP/Mのイメージファイルが作成できます。

HEXファイルは、ロードするアドレスを持っているので、ここで普通にDDTでロードすると、現在のBIOSの位置に読み込もうとしてしまいます。そこで、HEXファイルの読み込み

*6 CCPは「Console Command Procceser」、BDOSは「BASIC Disk Operating System」、BIOSは「BASIC I/O System」の略。これらはすべてDRI版CP/Mソフトウェア。
*7 CP/Mは評価目的/教育目的には無償で利用できます。詳細はCalderaのライセンスをご確認ください。

写真2　デバッグ中のシステム

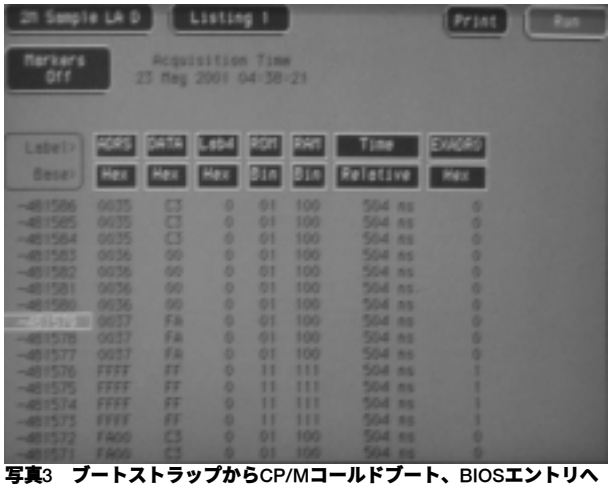


写真3　ブートストラップからCP/Mコールドブート、BIOSエントリへジャンプする瞬間をロジックスコープで表示したところ

のオフセットをかけられるように、DDTのRコマンド(Readコマンド)にオフセット分の数値を指定します。2の補数の数値を足したとき、本来のBIOSアドレスがTPA上のCP/MイメージのBIOSアドレスである「1F80H」になるように変換するので、

20K CP/Mの場合は　D580H
56K CP/Mの場合は　4580H
62K CP/Mの場合は　2D80H
64K CP/Mの場合は　5580H

のようにオフセットを指定します。例えば、BIOSのHEXファイルが「MYBIOS.HEX」である場合、64K CP/Mの場合は、BIOSをFA00Hからスタートするようにアセンブルし、

```
-IMYBIOS.HEX
-R2580
```

のように、2580Hのオフセット付きで読み込むことになります。読み込んだ後は、前と同じようにSAVEコマンドでディスク上に保存します。ちなみに、DDTから抜けるときには「Ctrl+C」と入力してください。すると、TPAのメモリイメージをそのままにしてCCPへ戻れるので、CCPのビルトインコマンドのSAVEでTPAのメモリイメージをディスクに吐き出すことができます。

ROMディスクにシステムを書き込む

さて、今回はROMディスクを利用することにしたのですが、Yazeのディスク管理プログラムcdmを使うと、ROMディスクのイメージファイルも簡単に作成できます。

先ほど、「CP/Mイメージファイルはcreateというコマンドで作成する」と書きました。このコマンドは、ディスクパラメータを自分で指定できるようになっています。ディスクパラメータ通りのCP/Mイメージファイルを用いるときには、128bytesのヘッダファイルさえ落とせば、そのままROMに焼くことができるバイナリファイルが得られます。焼き付ける前には、Yazeによってきっちりシミュレーションしておけば、後から焼き直しを行うようなことはほとんどなくなるでしょう。

我々は、研究室にあった1MbitのEPROMにROMディスクを作成するために、トラック容量8KbytesのROMディスクのイメージを作成することにしました(CP/Mとブートローダを1トラックに収めるため、8Kbytesがちょうどよい具合なのです)。cdmコマンドで作成したディスクイメージから先頭のヘッダを取り除き、シミュレーションのスクリプトを作成します。これらのファイルをPARTHENONシステムで読めるように、いろいろなツールで加工して、Linux上でPARTHENONのシミュレーションを実行しました。

PARTHENONシステムによる論理シミュレーションは、ROMからコールドブートして、システムをRAMに転送し、CP/Mに切り替え、システムレディが出た後に、MBASICで簡単な計算をさせるところまでいっています。ここに来るまでには、当然ながら何件かバグが見付かっています。参照した命令コード表自体のバグもあったのですが、これはCP/Mのシミュレーションで発見され、対策の確認はすでにできています(写真2、写真3)。

シリアルインターフェイス経由で駆動するので、シミュレーションも端末に相当するUARTを接続して、1文字ずつシステムに転送するスクリプトを使います。システムが出力す

る文字をモニタして、

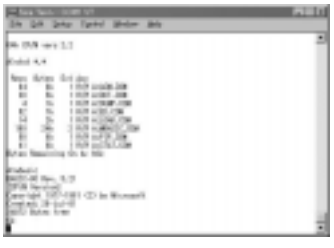
```
42 41 53 49 43 2d 38 30 20 52 65 76 2e 20 35 2e
32 31 0d 0a
```

と出てきたりするとちょっと感動ものです(分かります？)。

執筆時点では、まだ実機の最終確認に至っていません。原因は、ダウンロードケーブルの作成だったり、ROMライタの使い方だったり、ALTERAのライセンスファイルの到着遅れだったり、論理合成系のバグらしきものだったりとさまざまです。

一発目では動かなかったので、プッシュスイッチからマニュアルクロックを入れられるようなチャタリング防止回路と、クロック入力選択回路をALTERAのCPLDに入れて、手動クロックで動作を追っかけていくと、2バイト命令のBIOSの拡張命令部分が(OPコードは$DD)、どうも3バイト命令として認識されているようで「DD 02 23 ……」という命令の並びが3312番地へのコール命令($CD)に変わっているようです。そこで、論理合成結果のEDIFファイルをじっくり眺めていると、どうやらデータ線のbit4だけが負論理に合成されている。う〜ん、謎だ(コンパイラもそうだけれど、常に最新のツールを追っかけていると時々落し穴に落ちますね。もっとも、元のファイルに問題があるのかもしれませんから、現時点ではこれ以上は追っかけませんでした)。

このピンを反転するように論理変更したら、無事にBIOSのブートストラップが動作し始めました。いろいろやりながらCP/Mのローディングが終了しBIOSまでたどり着いていますが、まだ起動画面は出ていません。シリアルポートにラインアナライザを付けると、ちゃんと起動文字列が出ているし、アナライザから「DIR^M」と入力すれば、ちゃんとディレクトリが表示されているのであと一息なんですけど……(原稿執筆後、システムが動作したので、動作画面を画面1に載せておきます。ターミナルソフトと、我々のUARTのハードウェアハンドシェイクに用いるラインが違っていたことがトラブルの原因でした)。



画面1
動作画面

A 終わりに

これ以上は個別事情が色濃く出るので、このへんで終わりにしておきます。YazeとcdmのCP/Mのシステム開発が簡単に(？)できそうだ、という感触は得てもらえたでしょうか？ i8080A程度のCPUは、フリーでダウンロード可能な開発ソフト(MAX+PLUS2 Baseline)で開発できる程度の規模なので、「Yazeの上で高級言語で開発を行い、組み込み機器にCPLD版のCPUを組み込む」というのはいかがでしょうか？ CPUを自前で作れば、ちょっとした独自命令などで組み込みソフトも簡単になるし、何よりCPLDなら恐怖の「DISCON」に脅かされる必要はありません。駄目なら他のCPLDに論理ごと移してしまえばいいだけですからね。このCPUコアのソースコードは研究室のホームページに公開します([7])。

翻訳は遅々として進まないので、まだ皆さんにAlphaソフト開発の話を御紹介できません。申し訳ございませんが、もう少しお待ちください。

この夏、8/1〜8/3にかけて東海大で行われるパルテノン講習会の中で、この記事のシステムMY80の開発の講演とシステムのデモを行います。詳細は、パルテノン研究会のページに掲載されますので興味ある方はぜひ御参加ください。

RESOURCE

[1] パルテノン研究会
http://www.kecl.ntt.co.jp/parthenon

[2] ALTERA社CPLDのLinuxサポートに関するアナウンス
http://www.altera.com/corporate/press_box/releases/pr-linux_quartus.html

[3] Microsoft BASIC Version Information
http://www.emsps.com/oldtools/msbasv.htm

[4] Yet Another Z80 Emulatorの入手先
ftp://ftp.ping.de/pub/misc/emulators

[5] The unofficial CP/M web site.
http://www.retroarchive.org/cpm/archive/unofficial/

[6] Operating Systems and Board drivers(DRIPAKの入手先)
http://www.retroarchive.org/cpm/os/DRIPAK.ZIP

[7] My80 ——i8080A instruction compatible processor
http://shimizu-lab.et.u-tokai.ac.jp/pgm/my80/index.html