

OS 基礎学習のための 16BitOS の開発

目次

第一章 序論

- 1.1 オペレーティングシステムとは
 - 1.1.1 ハードウェアの抽象化とは
 - 1.1.2 リソースの管理
 - 1.1.3 コンピュータの利用効率の向上
- 1.2 研究目的

第二章 8086 の概要

- 2.1 CPU とは
- 2.2 x86 について
 - 2.2.1 x86 系の歴史
 - 2.2.2 プロテクトモードとは
 - 2.2.3 リアルモードとは
- 2.3 8086 について
 - 2.3.1 セグメントレジスタの特徴
- 2.4 レジスタとは
- 2.5 汎用レジスタ
- 2.6 フラグレジスタ
- 2.7 セグメントレジスタ
- 2.8 ポインタレジスタ

第三章 MikeOS と開発環境

- 3.1 MikeOS について
- 3.2 開発環境
- 3.3 実行方法
 - 3.3.1 シェルスクリプトとは
- 3.4 ソースコードのビルド
- 3.5 プログラムの実行

第四章 mikeos のディレクトリ構成

- 4.1 mikeos のディレクトリ構造について

第五章 bios コール詳細及びアプリケーションの実行結果

- 5.1 bios コールとは

第6章 まとめと展望

付録1 mikeos のソースコード解析

第一章

序論

本章の構成

- 1.1 オペレーティングシステムとは
 - 1.1.1 ハードウェアの抽象化とは
 - 1.1.2 リソースの管理
 - 1.1.3 コンピュータの利用効率の向上
 - 1.2 研究目的
-

1.1 オペレーティングシステムとは

オペレーティングシステムは、ハードウェアとソフトウェアの中間に位置する中間ソフトウェアである。主な目的はハードウェアの抽象化、リソースの管理、コンピュータの利用効率の向上にある。

1.1.1 ハードウェアの抽象化とは

コンピュータの製造元が異なるなどの理由で、実現する機能は同じだがハードウェアの詳細的な仕様に差異が生じることが多く存在する。そのためハードウェアの抽象化された利用方法を提供する。つまり、どのハードウェアでも同じ動作、利用方法を提供することでハードウェアの差異を無くし、アプリケーションソフトウェアの開発を容易にする。

1.1.2 リソースの管理

複数のアプリケーションを動作する場合、互に独立してアプリケーションを実行するにあたって必要なリソースを管理する。アプリケーションの実行の際にそれぞれのアプリケーションが、同じリソースを消費していた場合を競合といい、競合が起きた場合には、待たせる、エラーを返すなど、適切に対処する。

1.1.3 コンピュータの利用効率の向上

タスクとは並列実行の単位である。os はタスクを逐次的に実行し、タスクを消費する。複数のタスクを同時に実行する際に、様々な効率化を行い、コンピュータの単位時間当たりの処理能力であるスループットの向上を図る。一般的な用途では恩恵を感じることは比較的に少ないが、ウェブサーバやデータベースなどの大量のデータを処理するという分野にあたっては重要な項目となる。

1.2 研究目的

本研究の目的は、os の開発である。os の開発にあたる開発側と開発した os を使用するユーザー側の二つに分けて目標を定めている。開発側では簡易的で軽い os であること。分かりやすいソースコードであることを目的とし、ユーザー側では CPU の基礎を学ぶことに注力したアプリケーションの開発を目的としている。

第二章

8086 の概要

本章の構成

- 2.1 CPU とは
 - 2.2 x86 について
 - 2.2.1 x86 の歴史
 - 2.2.2 プロテクトモードとは
 - 2.2.3 リアルモードとは
 - 2.3 8086 について
 - 2.3.1 セグメントレジスタの特徴
 - 2.4 レジスタとは
 - 2.5 汎用レジスタ
 - 2.6 フラグレジスタ
 - 2.7 セグメントレジスタ
 - 2.8 ポインタレジスタ
-

2.1 CPU とは

CPU(Central Processing Unit)または中央演算処理装置はコンピュータの中心的な処理装置であり、コンピュータの頭脳や心臓部に例えられることが多い。CPU は記憶装置上にあるプログラムを逐次的に命令を読み込み、解釈、実行を行うことで情報の加工を行う。CPU は主記憶装置などのハードウェアをバスと呼ばれる信号線を介してデータまたはプログラムなどの情報のやり取りを行う。

2.2 x86 について

x86 とは Intel 8086 およびその互換性を持つマイクロプロセッサの命令セットアーキテクチャの総称である。プロセッサの型番が 8086、80186、80286、80386、80486 と続いたため、総称して 80x86 となり、下二桁が共通することから x86 または 86 系と呼ばれるようになった。

2.2.1 x86 の歴史

x86 の歴史は、世界最初のマイクロプロセッサ 4004 から、8 ビットの 8086 を経て、1978 年に発売された 16 ビットマイクロプロセッサ 8086 から始まる。セグメントと称されるアドレッシング法により 1MB という当時としては広大なメモリ空間をサポートしている。さらに別に I/O 空間も設けられている。1979 年には外部データバスを 8 ビットとした、8 ビット用の周辺 IC の利便性を図った 8088 を発表した。1982 年には 80186、80286 が発表された。この時になると CPU が実行できる操作の制限を行う CPU モードを設けていた。80286 はプロテクトモードや、24 ビットのアドレス空間を持つなどといった特徴があるが、パーソナルコンピュータでは自らが x86 互換の動作を行うモードであるリアルモードで、ほとんどが単に高速な 8086 としてしか活用されなかった。1985 年になると 32 ビットに拡張された 80386 が発表された。これは後に IA-32 と呼ばれるアーキテクチャである。32 ビット化にあたって、設計が見直され、大型コンピュータと渡り合えるような、という意味でコンピュータとして再設計された。80386 は 8086 ほぼそのままのリアルモードと、32 ビットのプロテクトモードを持つ。プロテクトモード中の際、仮想 86 モードによって従来の 8086 のプログラムを仮想化して実行可能である。その後、486、Pentium と続き、64 ビット化を目的とした x64 アーキテクチャを採用し、今日に至る。

2.2.2 プロテクトモードとは

プロテクトモードとは、80286 以降の x86 アーキテクチャの CPU モードの一つである。アプリケーションソフトウェアへの os の制御能力の向上を目的としており、正しい名称は Protected Virtual Address Mode (保護仮想アドレスモード) と呼ばれる。階層的な特権管理 (リングプロテクション) や、タスク間のメモリ保護 (プロテクト) を行うことが可能である。

2.2.3 リアルモードとは

リアルモードとは、x86 プロセッサの動作モードで、8086 互換の動作をする CPU モードである。全ての x86 プロセッサの起動時の動作モードであり、BIOS はこのモードで動作している。このモードでは、全てのレジスタのアドレス幅がデフォルトで 16 ビットであり、セグメントによる 20 ビットのアドレス空間にアクセスすることが可能である。

2.3 8086 について

8086 は 1978 年に発売された、Intel 社が開発した 16bitcpu である。8 ビットアーキテクチャである 8080 を 16 ビットに拡張し、乗除算などの命令を追加しており、アドレスバスは 20 ビットに、データバスは 16 ビットに拡張している。8086 のアーキテクチャの特徴として演算用のレジスタに加えて、セグメントレジスタというアドレス変換を目的としたレジスタを持っていることである。

2.3.1 セグメントレジスタの特徴

セグメントレジスタは、メモリ空間の拡張を目的としており、8086 は 64KB のアドレスをセグメントレジスタにより 1MB のメモリ空間を利用可能にできる。しかしながら、互換品または後継品が使用されるにつれて 64KB のメモリ空間は狭くなり、アプリケーションのプログラム自体が自力でセグメントレジスタを操作することもあり、プログラマ側がセグメントレジスタを頻繁に操作するため、プログラミング上の不便性があり、セグメント方式は批判を浴びることになった。だが、互換性を重視しつつ開発が短期間で完了できるという点から、コストパフォーマンスに優れた方式であった。

2.4 レジスタとは

論理回路において、フリップフロップなどにより状態を保持する装置である。コンピュータにおいて、プロセッサが内蔵している部分を指す。命令セットで明示的に操作するレジスタの他にプロセッサ自身が動作するための内部レジスタと呼ばれるレジスタがある。プロセッサ内部のレジスタは、計算結果の保持、RAM または ROM などのメインメモリにアクセスする際のアドレスの保持、プロセッサや周辺機器の動作状態を保持・変更を行う。本研究では 8086 を想定したオペレーティングシステムの開発を行う。そのために 8086 のレジスタの種類を示す。

2.5 汎用レジスタ

一般的な算術命令、カウンタ、ビット演算、データ転送などで使用可能なレジスタ群。16ビットのレジスタだが、8ビットずつ分けることができる。その場合、○Hレジスタと○Lレジスタに分類される。汎用レジスタの概要を表2.1に示す。

表 2.1 汎用レジスタ

名称	内容
AX(Accumulator Register)	一般的に算術演算で使われるレジスタ。累計を行うのも可能。乗算命令、除算命令でも使われる。また、ポート入出力命令で、データを格納するために使われる。
BX(Base Register)	ポインタレジスタとして使用可能なレジスタ。ポインタレジスタとは、アドレスを設定してメモリにアクセスすることが可能なレジスタ。
CX(Count Register)	繰り返し命令で、暗黙的にカウンタとして使われるレジスタ。
DX(Data Register)	乗算命令、除算命令などで使われるレジスタ。ポート入出力命令で、256番地以上のポートアドレスを指定するときに使用する。

2.6 フラグレジスタ

cpu の内部状態を表す 16 ビットのレジスタである。実際に使用されるのは 9 ビットで制御フラグと状態フラグに大別することができる。表 2.2 に制御フラグの概要を、表 2.3 に状態フラグの概要を示す。制御フラグは、cpu の動作に影響を及ぼすものであり、フラグの値が異なれば、同じ cpu 命令でも、異なる動作を行う。状態フラグは、桁上がりやパリティなどの cpu 命令を実行した結果による付加的な情報が反映される。

表 2.2 フラグレジスタ 制御フラグ

名称	内容
DF(Direction Flag)	方向フラグ。連続したメモリアクセス時にアドレスを加算するか減算するかを決定する。
IF(Interrupt Enable Flag)	割り込み許可フラグ。マスク可能な割り込みの制御を行う。
TF(Trap Flag)	トラップフラグ。1 つの命令ごとに割り込みを発生させるときに使用される

表 2.3 フラグレジスタ 状態フラグ

名称	内容
OF(Overflow Flag)	オーバーフローフラグ。算術演算の結果が有効なビット幅で収まらなかったときにセットされる。
SF(Sign Flag)	サインフラグ。演算結果の最上位ビットがセットされる。
ZF(Zero Flag)	ゼロフラグ。演算結果がゼロのときにセットされる。
AF(Auxiliary Carry Flag)	補助キャリーフラグ。BCD 演算で利用。BCD 演算では 4 ビットで 0 から 9 までの演算を行うため、4 ビットの最上位ビットで発生したキャリー（桁上がり）またはボロー（桁借り）がセットされる。
PF(Parity Flag)	パリティフラグ。演算結果の最下位バイトに 1 のビットが偶数個あるときにセットされる。
CF(Carry Flag)	キャリーフラグ。算術演算でキャリーまたはボローが発生したときにセットされる。

2.7 セグメントレジスタ

セグメントとは分割されたメモリの一部を指すものであり、メモリ空間を拡張する方法である。セグメントの開始位置を指定する専用のレジスタがセグメントレジスタとなる。表 2.4 にセグメントの概要を示す。

表 2.4 セグメントレジスタ

名称	内容
CS(Code Segment)	プログラムの実行セグメントを表す。IP レジスタを使用したメモリアクセス時に参照される
DS(Data Segment)	データを参照する時のデフォルトセグメント。SI または DI レジスタを使用したメモリアクセス時に参照される。
ES(Extra Segment)	異なるセグメント間のコピーなどで使用される。DI レジスタを使用したメモリアクセス時に参照される。
SS(Stack Segment)	スタックポインタを使用するときに参照される。SP または BP レジスタを使用したメモリアクセス時に参照される。

2.8 ポインタレジスタ

cpu がアドレスをしてするときを使用することができるレジスタである。表 2.5 にポインタレジスタの概要について示す。一部の転送命令で、転送元または転送先として、暗黙的に利用される場合がある。

表 2.5 ポインタレジスタ

名称	内容
SI(Source Index)	メモリの転送命令などで、転送元アドレスとして利用される。また、デフォルトで DS レジスタを参照する。
DI(Destination Index)	メモリの転送命令などで、転送先アドレスとして利用される。また、デフォルトで ES レジスタを参照する。
BP(Base Pointer)	局所的な変数を参照する。スタックポインタとして使用可能なレジスタ。必ず SS レジスタを参照する。
SP(Stack Pointer)	スタックポインタとして使用可能なレジスタ。必ず SS レジスタを参照する。
IP(Instruction Pointer)	次に実行する命令のアドレスを示すレジスタ。IP レジスタの変更は、直接値を設定するのではなく、分岐命令や関数呼び出し命令などで、行われる。必ず CS レジスタを参照する。

第三章

MikeOS の概要

本章の構成

- 3.1 MikeOS について
 - 3.2 開発環境
 - 3.3 実行方法
 - 3.3.1 シェルスクリプトとは
 - 3.4 ソースコードのビルド
 - 3.5 プログラムの実行
-

3.1 MikeOS について

MikeOS はアセンブリ言語で記述されたオペレーティングシステムである。単純な 16 ビットのリアルモードがどのようにして機能するかを目的とした学習用のオペレーティングシステムであり、ソースコードが軽く、オープンソースである。本研究で開発したオペレーティングシステムはこの MikeOS を元に開発した。この章では開発したオペレーティングシステムのコンパイル及び実行に必要な開発環境について示す。

3.2 開発環境

本研究の開発環境を表 3.1 に示す。オペレーティングシステムの開発には Ubuntu を主に使用し、ソースコードの編集及びコンパイルを行う。Ubuntu を採用する理由は、MikeOS のディレクトリ内にあるビルド、テストを行うシェルスクリプトを使用するためである。ビルドを行うシェルスクリプトはコンパイルの他にディスクイメージを作成する。Windows でもバッチファイルを実行することでコンパイルを行うことは可能である。しかしながら、ディスクイメージを作成するにあたって必要なプログラムが Windows10 では実行が不可能であったため、今回の研究では利便性という点から Ubuntu を採用した。Ubuntu は Windows から仮想化ソフトウェアである Virtual Box を使用して仮想化させて実行している。この Ubuntu から MikeOS を実行させる場合、qemu と呼ばれる、Virtual Box とは別の仮想化ソフトウェアを Ubuntu から実行し、起動させる。

表 3.1 開発環境

内容	項目
Windows10	仮想化用 OS
Virtual Box	仮想化ソフトウェア
Ubuntu 20.04.3 LTS	仮想化で使用する OS
Visual Studio Code 1.62.2	コードエディタ
nasm x86 syntax highlighting v1.2.0	vscode の拡張機能
nasm 2.15.02-1 amd64	コンパイルを行うためのアセンブラ
qemu 1:4.2-3ubuntu3.18 amd64	仮想化ソフトウェア
MikeOS 4.6.1	本研究で使用する OS

3.3 実行方法

開発するオペレーティングシステムのビルド、実行には MikeOS のディレクトリ内にあるシェルスクリプトを使用する。表 3.2 に使用するシェルスクリプトを示す。

表 9 シェルスクリプト一覧

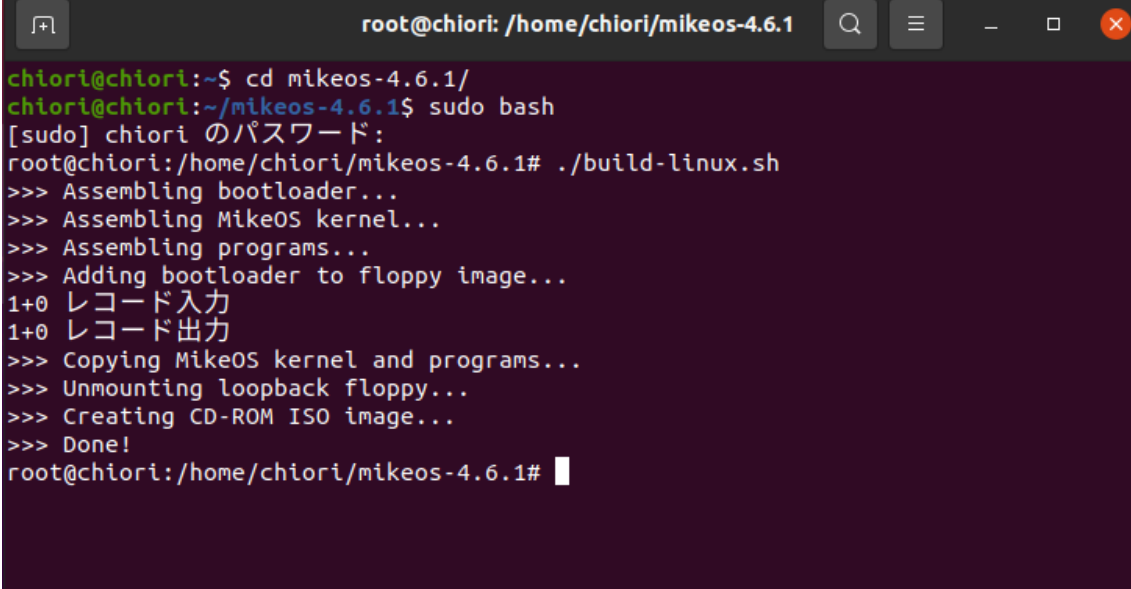
スクリプト名	ファイルパス
build-linux.sh	mikeos-4.6.1/build-linux.sh
test-linux.sh	mikeos-4.6.1/test-linux.sh

3.3.1 シェルスクリプトとは

シェルとはユーザーのためにインターフェイスを提供するプログラムである。コンピュータシステムとユーザーの間にある殻であることからシェルと呼ばれている。オペレーティングシステムが提供する機能へのアクセスの提供などを行う。シェルスクリプトはコマンドラインによる命令をひとまとめにし、命令を連続的に実行する。windows の場合はバッチファイルと呼ばれ、Unix 系の場合にはシェルスクリプトと呼ばれており、それぞれ命令に違いがある。本実験ではシェルスクリプトを使用する。

3.4 ソースコードのビルド

MikeOS のビルドには `build-linux.sh` を実行する。手順としては、Ubuntu の端末(ターミナル)から実行する。端末を起動し、MikeOS のディレクトリに移動した後、`sudo bash` コマンドを実行し、シェルを起動させる。そして、`build-linux.sh` を実行することでソースコードにエラーが無ければビルドされる。このプログラムはコンパイルを行うと同時に iso ファイルも作成される。図 3.1 に `build-linux.sh` の実行結果を示す。

A terminal window titled 'root@chiori: /home/chiori/mikeos-4.6.1' with standard window controls. The terminal shows a user 'chiori' at 'chiori' prompt, navigating to the 'mikeos-4.6.1' directory and running 'sudo bash'. After entering the password, the user runs './build-linux.sh'. The script outputs several steps: 'Assembling bootloader...', 'Assembling MikeOS kernel...', 'Assembling programs...', 'Adding bootloader to floppy image...', 'Copying MikeOS kernel and programs...', 'Unmounting loopback floppy...', and 'Creating CD-ROM ISO image...'. It also shows disk activity for 'レコード入力' (recording input) and 'レコード出力' (recording output). The process ends with 'Done!' and the prompt returns to 'root@chiori:/home/chiori/mikeos-4.6.1#'.

```
root@chiori: /home/chiori/mikeos-4.6.1
chiori@chiori:~$ cd mikeos-4.6.1/
chiori@chiori:~/mikeos-4.6.1$ sudo bash
[sudo] chiori のパスワード:
root@chiori:/home/chiori/mikeos-4.6.1# ./build-linux.sh
>>> Assembling bootloader...
>>> Assembling MikeOS kernel...
>>> Assembling programs...
>>> Adding bootloader to floppy image...
1+0 レコード入力
1+0 レコード出力
>>> Copying MikeOS kernel and programs...
>>> Unmounting loopback floppy...
>>> Creating CD-ROM ISO image...
>>> Done!
root@chiori:/home/chiori/mikeos-4.6.1#
```

図 3.1 `build-linux.sh` の実行結果

3.5 プログラムの実行

開発したオペレーティングシステムの実行及び、テストを行うには test-linux.sh を実行する。このシェルスクリプトはビルド時に作成されるディスクイメージを参照し、qemu によって実行される。図 3.2 に test-linux.sh の実行前を、図 3.3 に test-linux.sh の実行後を示す。

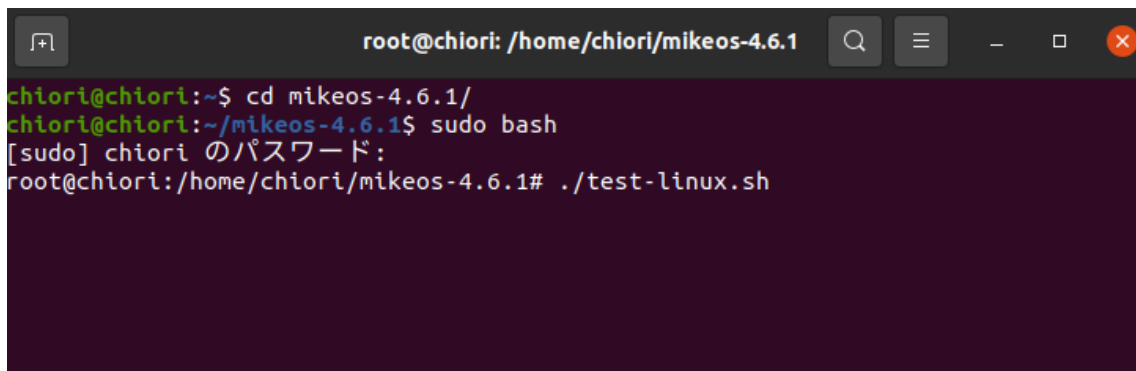


図 4 test-linux.sh の実行前



図 5 test-linux.sh の実行後

第四章

MikeOS のディレクトリ構成

本章の構成

- 4.1 MikeOS のディレクトリ構成について
 - 4.2 汎用レジスタ
 - 4.3 フラグレジスタ
 - 4.4 セグメントレジスタ
 - 4.5 ポインタレジスタ
-

4.1 MikeOS のディレクトリ構成について

この章では、MikeOS の構造について示す。まず MikeOS の全体的なディレクトリ構成を図 4.1 に、ディレクトリの概要を表 4.1 に示す。

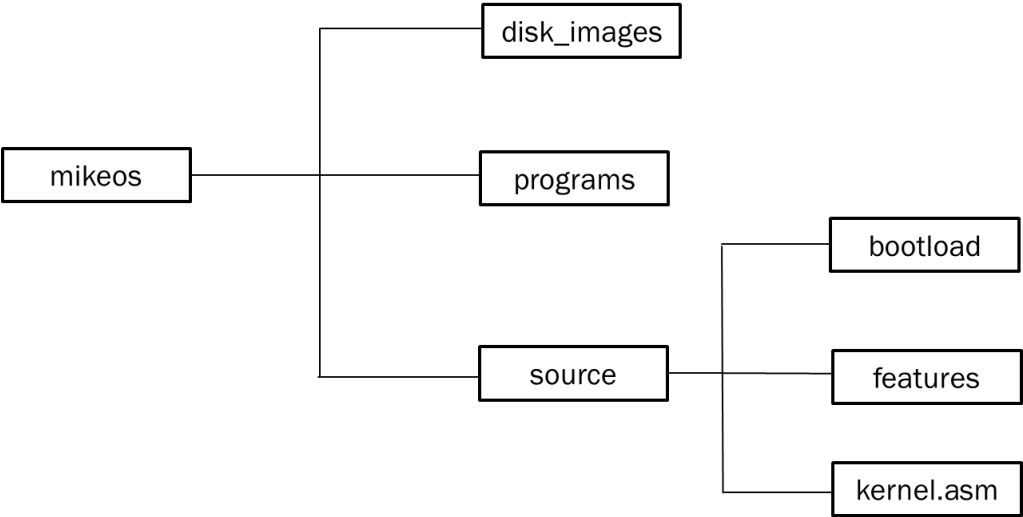


図 4.1MikeOS のファイル構成

表 4.1MikeOS のディレクトリ概要

ディレクトリ名 プログラム名	概要
disk_images	ビルド時に生成されるディスクイメージを格納するディレクトリ
programs	アプリケーションのソースコードのディレクトリ
source	os のソースコード全体のディレクトリ
source/bootload	BOOTLOAD.BIN を生成するソース。ビルド時にディスクイメージに追加
source/features	FAT12 サポート、文字列ルーチン、BASIC インタープリターなどの MikeOS のコンポーネント
source/kernel.asm	他のソースファイルをプル(取り込む)するコアカーネルソースファイル

4.2 mikeOS/ の概要

ディレクトリ MikeOS の下のディレクトリ内容を示す。MikeOS/にはビルド、qemu による MikeOS の実行を行うシェルスクリプトまたはバッチファイル、ドックストリングなどが含まれている。表 4.2 に MikeOS/のディレクトリ内容について示す。

表 4.2 mikeOS/のディレクトリ概要

ディレクトリ名 スクリプト名	概要
disk_images	ビルド時に生成されるディスクイメージを格納するディレクトリ
doc	ドックストリング
programs	アプリケーションのソースコードのディレクトリ
source	os のソースコード全体のディレクトリ
README.TXT	MikeOS に関する説明、ライセンス
README.md	README.TXT の md 版
build-linux.sh	ソースコードのビルド、 ディスクイメージの作成を行うシェルスクリプト
build-openbsd.sh	build-linux.sh の openbsd 版
build-osx.sh	build-linux.sh の mac osx 版
buildwin.bat	build-linux.sh の windows 版
test-linux.sh	ディスクイメージの実行を開始行うシェルスクリプト

4.2 mikeOS/programs の概要

ディレクトリ名 programs は、オペレーティングシステム実行後のアプリケーション部分になる。MikeOS 実行後に programs に格納されたプログラムを任意で実行できるようになる。このディレクトリ内に開発したプログラムを格納し、ビルドにエラーが無ければ、そのアプリケーションを含んだディスクイメージが生成される。MikeOS は BASIC 言語のインタプリタを持っているため、BASIC 言語のアプリケーションも作成可能である。表 4.4 に programs の概要について示す。ディレクトリ内には拡張子.bin が存在するが、これは拡張子.asm がコンパイルに作成されるバイナリファイルである。意味合いは.asm と同じであるため省略する。

表 4.3 mikeOS/programs のディレクトリ概要

プログラム名	概要
adventure.bas	テキストアドベンチャーゲーム
archive.bas	コマンドライン形式のファイルアーカイビングプログラム
calc.bas	計算機プログラム
cf.bas	宇宙飛行ゲーム
draw.bas	ASCII アート描画プログラム
edit.asm	フルスクリーンテキストエディタ
example.bas	BASIC 機能のデモンストレーション
fileman.asm	フロッピーディスク上のファイルを削除、名前変更、コピー
fisher.asm	釣りゲーム
forth.asm	Forth システムプログラム
hangman.asm	都市の名前当てゲーム
keyboard.asm	キーボード入力によるサウンド出力プログラム
mbpp.bas	BASIC 用の追加ライブラリプログラム
memedit.bas	メモリ管理プログラム
mikdev.inc	call 関数呼び出しのためのインクルードファイル
monitor.asm	障害物除けゲーム
muncher.bas	スネークゲーム
sample.pcx	viewer.asm のための画像
serial.asm	シリアル端末プログラム
sudoku.bas	数独ゲーム
viewer.asm	テキストファイル、SAMPLE.PCX などの 320*200*16PCX 画像の表示

4.3 本研究の programs のディレクトリ概要について

本研究の開発したオペレーティングシステムはMikeOSに存在するアプリケーションを使用しないため、インクルードファイル以外を削除している。programs 内には三つのアプリケーションを追加した。表 4.3 に追加したアプリケーションを示す。

表 4.3 本研究の programs のディレクトリ概要

プログラム名	概要
int10.asm	BIOS コール int10 の実行を行うプログラム
int16.asm	BIOS コール int16 の実行を行うプログラム
int1a.asm	BIOS コール int1A の実行を行うプログラム
mikedevelop.inc	call 関数呼び出しのためのインクルードファイル

4.3.1 mikedevelop.inc について

mikedevelop.inc は、関数のインクルードの際に入力するディレクトリのパスを省いたものと考えられる。外部の関数を使用するために、関数のインクルードを使用する。使用するにあたって、関数のインクルードにはパスの入力が必要であるが、プログラムを機能ごとに分割すればするほど、インクルードするファイル数が増加してしまう。mikedevelop.inc は、その不便性を解消したものであり、mikedevelop.inc をインクルードするだけで外部関数である call 関数がほとんど使用可能になる。

4.4 mikeOS/source の概要

mikeOS/source はオペレーティングシステムの根幹をなすプログラム部分である。表 4.4 に mikeOS/source のディレクトリ内容の表を示す。

表 4.4 本研究の programs のディレクトリ概要

ディレクトリ名 プログラム名	概要
bootload	オペレーティングシステム起動時に実行するプログラム
features	FAT12 サポート、文字列ルーチン BASIC インタプリタなどの MikeOS のコンポーネント
kernel.asm	オペレーティングシステムの主体となるカーネルプログラム

4.4 mikeOS/source/bootload.asm の概要

オペレーティングシステムのソースコードは膨大のため、少しずつプログラムをロードしていく。pc 電源投入時に、BIOS により起動プログラムをロードする。この起動プログラムが bootload.asm にあたる。このプログラムはブートコード (ブートプログラム) といわれ、ユーザー側が作成する。bootload.asm はハードウェアのチェックが主な役割であり、エラーが無ければ、カーネルへとジャンプする。

4.4.1 電源投入時の処理

図 4.2 に電源投入時に行われる処理のフローチャートを示す。電源投入時には post(power of self test)と呼ばれる最低限のハードウェアチェックが行われる。エラーが無い、もしくは BIOS 設定に入る特定のキーを押下していない場合、起動装置として設定された外部記憶装置の先頭から 512 バイトを読み込んで、メモリの 7c00 番地にロードする。

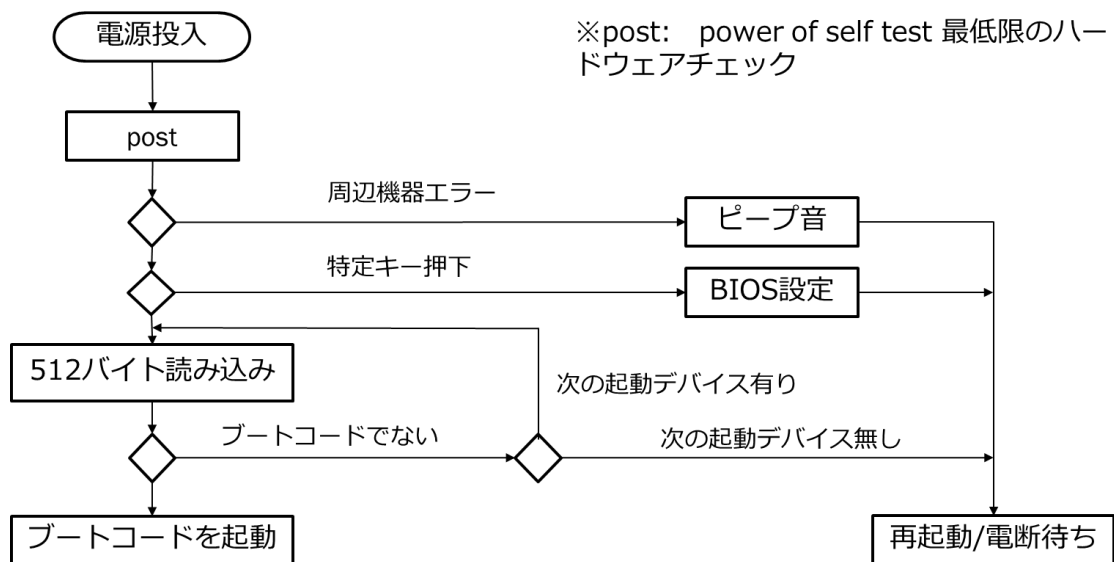


図 4.2 電源投入時に行われる処理のフローチャート

4.4 mikeOS/source/features の概要

ディレクトリ名 features は、MikeOS のコンポーネントが格納されたディレクトリである。このディレクトリ内にあるプログラムは単体で動作するプログラムは少なく、関数を外部で使用する目的で使用される。簡単に言えば、外部で使用される関数群といえる。この関数は CALL 関数といい、割り込みによる関数呼び出しになる。features ディレクトリ内には FAT12 サポート、文字列ルーチン、BASIC インタプリタなどのコンポーネントが備わっている。表 4.5 に features のディレクトリ内容について示す。

表 4.5 本研究の programs のディレクトリ概要

プログラム名	概要
basic.asm	BASIC 言語のインタプリタ
cli.asm	ターミナルプログラム
disk.asm	ハードディスク操作中心のプログラム
keyboard.asm	キーボード操作中心のプログラム
math.asm	進数変換、乱数取得などの数値を扱うプログラム
misc.asm	エラー文、待機を実行するプログラム
ports.asm	ポートによる取得、送信を行うプログラム
screen.asm	画面出力を行うプログラム
sound.asm	音声を発生させるプログラム
string.asm	文字列操作を行うプログラム

4.5 mikeOS/source/kernel.asm の概要

kernel.asm は、bootload.asm が正常に作動した場合にロードされるプログラムである。このカーネル部分がこの MikeOS を動作させるプログラムである。このプログラムは、常に動作し続ける。programs のアプリケーションの任意実行、ハードウェアの管理はこの kernel により実行される。

4.5.1 プログラムの任意実行の仕組み

図 3 に kernel.asm の安定した動作を示した簡易的な図 4.3 を示す。これはカーネルがエラーを吐かずに正常に動作した場合のプログラムの実行例である。プログラムの実行を司る関数は app_selector である。app_selector 関数のチェックでエラーが無ければ、execute_bin_program 関数に移動する。execute_bin_program をはモニターの背景をクリアにし、選択した programs に含まれるプログラムを実行する。プログラムの実行時には call 32768 という call 命令で実行できる。32768 は MikeOS のメモリマップにおける外部プログラム用の 32K スペースである。つまり、このスペースに外部プログラムをロードさせたということになる。図 4.4 に MikeOS のメモリマップについて示す。

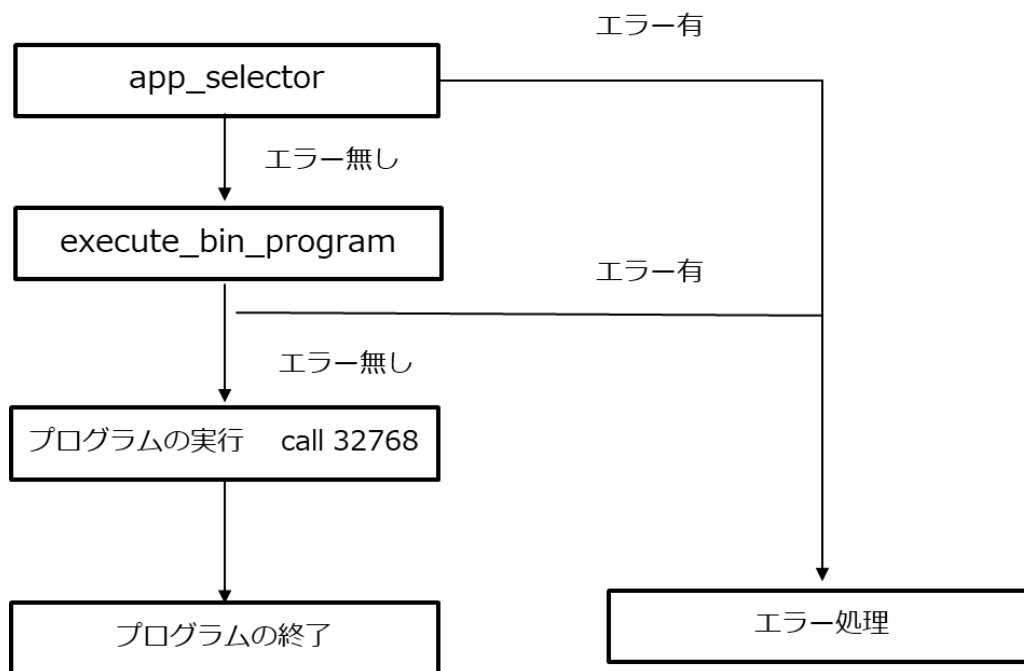


図 4.3 簡易的な kernel のプログラム実行の仕組み

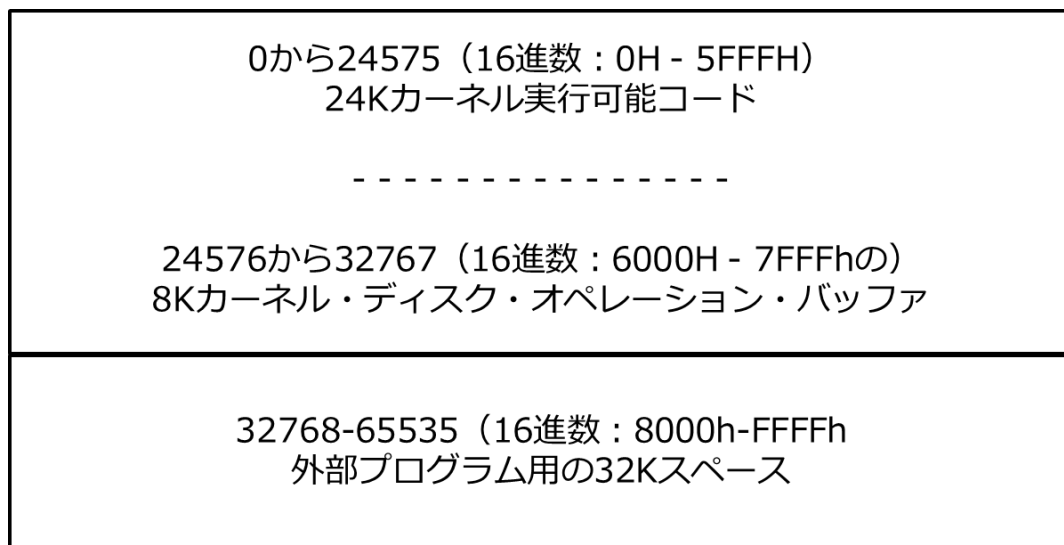


図 4.4 MikeOS のメモリマップ

第五章

bios コールの詳細及びアプリケーションの実行結果

本章の構成

- 5.1 bios コールについて
 - 5.2 汎用レジスタ
 - 5.3 フラグレジスタ
 - 5.4 セグメントレジスタ
 - 5.5 ポインタレジスタ
-

5.1 bios コールについて

bios コールは、コンピュータ本体に組み込まれたプログラムであり、多くのサービスプログラムを持つ関数群である。主に pc の電源投入後のハードウェアチェックに使用される。本実験では bios コールを実行するアプリケーションを三つ追加している。四章の 4.3 の表がこのアプリケーションに該当する。アプリケーションの目的は、汎用レジスタの理解、ハードウェア制御を目的としている。表 5.1 に bios コールの命令セットの詳細を示す。

表 5.1 本研究の programs のディレクトリ概要

割り込み番号	サービス名	概要
0x10	Video Services	画面出力
0x11	Equipment List Service	システムのデバイス情報の取得
0x12	Low Memory Size Service	1MB 以下の物理メモリ領域の メモリサイズを取得
0x13	Disk Services	ハードディスクの操作
0x14	Serial Port Services	シリアルポートの操作
0x15	General System Services	システムデバイスの要求と処理
0x16	Keyboard Services	キーボードの制御
0x17	Parallel Port Services	パラレルポートの制御
0x18	Boot Fault Routine	os 起動失敗時に行う処理
0x19	Bootstrap Routine	ブートローダーをロード
0x1A	Time/Date Services	時刻、日付を取得
0x1B	Control-Break Signal	ctrl キーと break キーの同時押しを行 った場合に作動
0x1C	User Timer Interrupt	タイマ割り込み
0x1D	Video Parameter Table	6845CRT 制御レジスタの設定
0x1E	Floppy Diskette Parameter Table	フロッピーディスクの制御
0x1F	Video Graphics Character Table	フォント変更時に使用
0x41/0x46	Fixed Disk Parameter Tables	IDE のプライマリまたは セカンダリドライブに使用する
0x4A	Real Time Software Interrupt	リアルタイムクロック割り込み
0 x 4 F	Video Bios Extension Function	VBE ファンクションの実行

5.2 int 命令

bios コールの実行を行うには、int 命令を使用する。アセンブリ言語による記述方法は、int 割り込み番号と記述する。例として、0x10 のビデオサービスを実行する場合、int10h と記述すればビデオサービスが実行可能になる。h はサフィックスによる 0x を表す。しかし、まだ、実行できたわけではない。複数の機能に別れているため、サービスの選択とともに機能の選択を行わなければならない。機能の選択は AL レジスタに番号を格納することで可能になる。

5.2 アプリケーションの構成

アプリケーション int10.asm、int16.asm、int1A のプログラム実行の際、初期に表示される選択画面を図 5.1 に、キーボード入力によって選択したファンクションを図 5.2 に示す。図 5.2 は、実行例として、プログラム int10、キーボード f が押された場合の結果である。bios コール名を記載したバー、命令に必要なレジスタの表、追加機能の表示の三つの画面で構成されている。この場合の追加機能はキーボード 1 の入力によって命令の実行を行い、Esc キーで選択画面に移動する。

図 5.1 アプリケーション実行時の選択画面

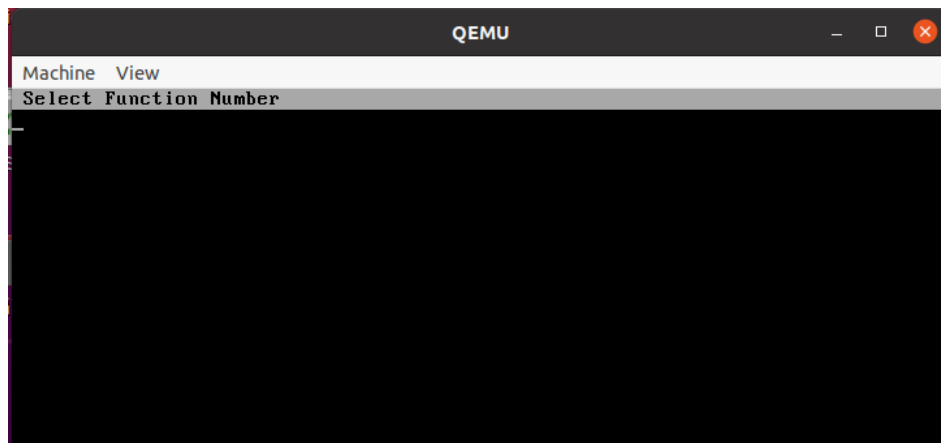


図 5.2 int10 キーボード f の実行結果

BIOSコール名

命令に必要なレジスタの表

追加機能
キーボード1:命令の実行
Escキー:選択画面に移動

```
Machine View
Write Teletype Mode
function number AH = 0x0E Write Teletype Mode
register  value  explanation
AL      **      set write character code
BL      0-15    set foreground color(only graphic mode)
BH      **      set videopage number

Please push keyboard
keyboard 1 write teletype mode
example AL=97 BL=0 BH=0
keyboard esc return select mode
```

5.3 bios コール 0x10

本研究の追加プログラムは、bios コール 0x10、0x16、0x1A の動作を実行可能である。bios コール 0x10 ビデオサービスの概要を表 5.2 に示す。表 5.2 は 0x10 によって実行可能なサービスの種類である。

表 5.2 0x10 ビデオサービスの概要

機能番号	ファンクション名	概要
0x00	Set Video Mode	ビデオモードの設定
0x01	Set Cursor Size	カーソルサイズの設定
0x02	Set Cursor Position	カーソルの位置の設定
0x03	Return Cursor Position	カーソルの位置を取得
0x04	Return Light Pen Condition	ライトペンの位置を取得 (通常の bios では何もしない)
0x05	Set Current Video Page	ビデオバッファのページ設定
0x06	Scroll Up Region	スクリーンの上方向スクロール
0x07	Scroll Down Region	スクリーンの下方向スクロール
0x08	Return Character and Attribute	文字コードとアトリビュートの取得
0x09	Write Character and Attribute	文字コードとアトリビュートの書き込み
0x0A	Write Character	文字コードの書き込み
0x0B	Set Color Palette	カラーパレットの設定
0x0C	Wtite Graphic Pixel	色情報の書き込み (通常の bios サービスでは無い)
0x0D	Read Graphic Pixel	色情報の読み込み (通常の bios サービスでは無い)
0x0E	Write Teletype Mode	文字コードの書き込み
0x0F	Return Video Display Mode	ビデオモードの取得
0x13	Write String	文字列の書き込み

5.4 アプリケーション int10.asm について

アプリケーションは、表 5.2 のビデオサービスを実行可能である。アプリケーション起動を起動すれば選択画面に入る。選択画面からキーボードの出力で、ビデオサービスのファンクションを選択できる。表 5.3 に割り当てたキーボードの詳細を示す。なお、ビデオサービスには通常の bios では何も動作しない命令がある。その命令には(不可)とし、命令の実行は行わない。

表 5.3 0x10 ビデオサービスの概要

ファンクション名	割り当てたキーボード
Set Video Mode	1
Set Cursor Size	2
Set Cursor Position	3
Return Cursor Position	4
Return Light Pen Condition(不可)	5
Set Current Video Page	6
Scroll Up Region	7
Scroll Down Region	8
Return Character and Attribute	9
Write Character and Attribute	a
Write Character	b
Set Color Palette	c
Write Graphic Pixel(不可)	d
Read Graphic Pixel(不可)	e
Write Teletype Mode	f
Return Video Display Mode	g
Write String	h

5.4.1 Set Video Mode

Set Video Mode は、ビデオモード設定である。ビデオコントローラのビデオモードレジスタに指定したビデオモードを設定する。ビデオモード設定を行った場合、スクリーンがクリアされてカーソル位置が一番左上になる。キーボード 1 Set Video Mode の初期画面を図 5.3 に示す。ビデオモード設定は追加機能は選択画面の遷移のみである。

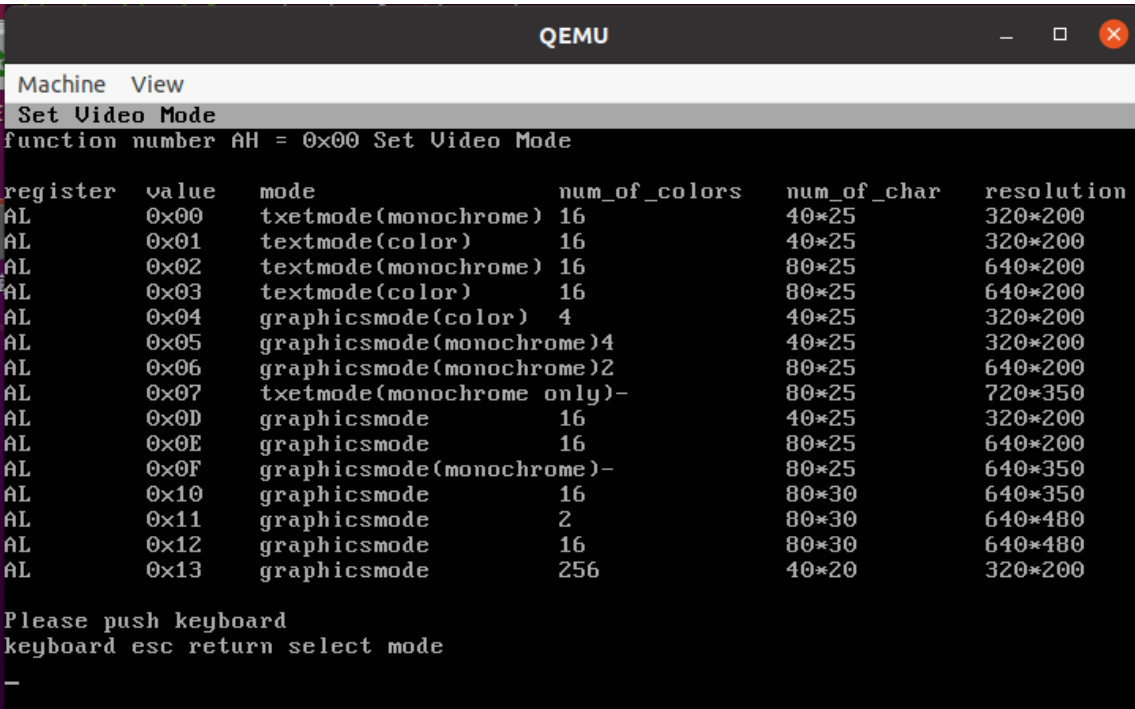


図 5.3 int10 キーボード 1 の初期画面

5.4.2 Set Cursor Size

Set Cursor Size は、カーソルサイズ設定である。この命令を呼び出すことで、テキストモードのカーソルサイズを設定する。この命令は選択画面から、キーボード 2 を入力することで実行可能になる。図 5.4 にキーボード 2 の初期画面を示す。追加機能はキーボード 1 で CH レジスタのインクリメント、キーボード 2 で CL レジスタのインクリメントを行う。CH レジスタはカーソルが始まるスキャンラインの設定。CL レジスタはカーソルが終わるスキャンラインの設定である。表 5.4 に追加機能の概要を、図 5.5 にキーボード 2 を五回押下した場合の実行結果を、その後キーボード 1 を三回押下した場合の実行結果を図 5.6 に示す。

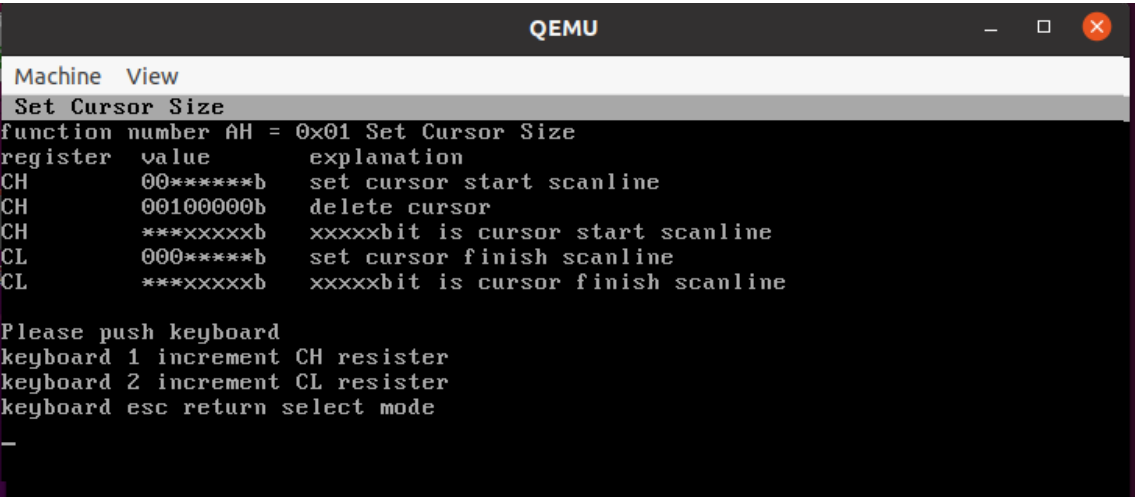


図 5.4 int10 キーボード 2 の初期画面

表 5.4 int10 キーボード 2 追加機能の概要

追加機能に割り当てたキーボード	詳細
1	カーソルの上部位置を変更
2	カーソルの下部位置を変更


```
QEMU
Machine View
Set Cursor Size
function number AH = 0x01 Set Cursor Size
register value explanation
CH 00*****b set cursor start scanline
CH 00100000b delete cursor
CH *****b xxxxxbit is cursor start scanline
CL 000*****b set cursor finish scanline
CL *****b xxxxxbit is cursor finish scanline

Please push keyboard
keyboard 1 increment CH resister
keyboard 2 increment CL resister
keyboard esc return select mode
■
```

図 5.5 int10 キーボード 2 追加機能の実行結果(キーボード 2 を五回押下)

```
QEMU
Machine View
Set Cursor Size
function number AH = 0x01 Set Cursor Size
register value explanation
CH 00*****b set cursor start scanline
CH 00100000b delete cursor
CH *****b xxxxxbit is cursor start scanline
CL 000*****b set cursor finish scanline
CL *****b xxxxxbit is cursor finish scanline

Please push keyboard
keyboard 1 increment CH resister
keyboard 2 increment CL resister
keyboard esc return select mode
■
```

図 5.6 int10 キーボード 2 追加機能の実行結果(キーボード 1 を三回押下)

5.4.3 Set Cursor Position

Set Cursor Position は、カーソル位置設定ファンクションである。命令実行時に、テキストモードのカーソル位置(X,Y)を設定することができる。X は列で、Y は行であり、それぞれ 0 から始まる。この命令は、選択画面からキーボード 3 を入力することによって実行可能になる。図 5.7 にキーボード 3 の初期画面を示す。追加機能はキーボード 1 で DH レジスタにインクリメント、2 で DL レジスタにインクリメント、キーボード 3 で DH レジスタにデクリメント、4 で DL レジスタにデクリメントを行う。追加機能の詳細を表図 5.8 にキーボード 1、図 5.9 にキーボード 2、図 5.10 にキーボード 3、図 5.11 にキーボード 4 を一回押下した場合の実行結果を示す。

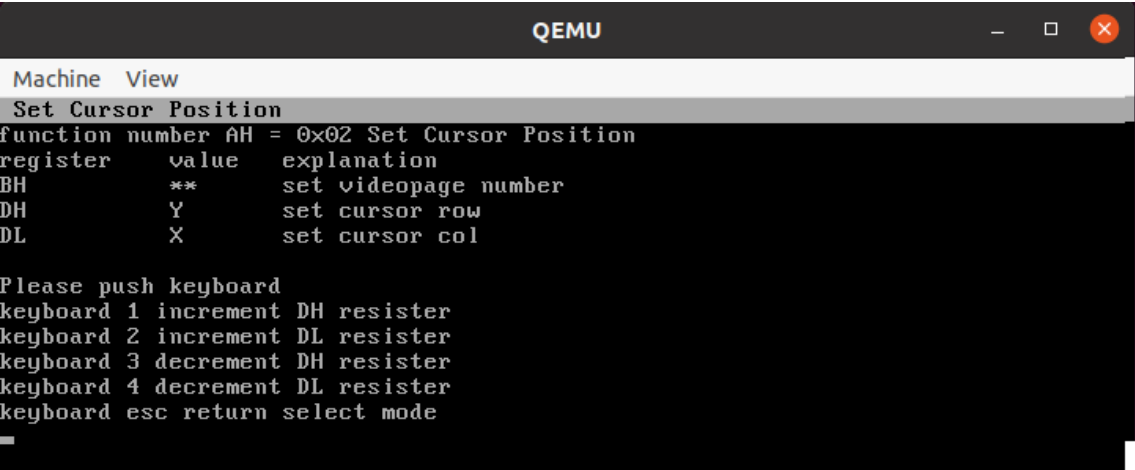


図 5.7 int10 キーボード 3 の初期画面

表 5.5 int10 キーボード 3 追加機能の概要

追加機能に割り当てたキーボード	詳細
1	一行下がる
2	一列右にずれる
3	一行上がる
4	一列左にずれる

```
QEMU
Machine View
Set Cursor Position
function number AH = 0x02 Set Cursor Position
register    value    explanation
BH         **       set videopage number
DH         Y        set cursor row
DL         X        set cursor col

Please push keyboard
keyboard 1 increment DH resister
keyboard 2 increment DL resister
keyboard 3 decrement DH resister
keyboard 4 decrement DL resister
keyboard esc return select mode
=
```

図 5.8 int10 キーボード 3 追加機能の実行結果(キーボード 1)

```
QEMU
Machine View
Set Cursor Position
function number AH = 0x02 Set Cursor Position
register    value    explanation
BH         **       set videopage number
DH         Y        set cursor row
DL         X        set cursor col

Please push keyboard
keyboard 1 increment DH resister
keyboard 2 increment DL resister
keyboard 3 decrement DH resister
keyboard 4 decrement DL resister
keyboard esc return select mode
=
```

図 5.9 int10 キーボード 3 追加機能の実行結果(キーボード 2)

```
QEMU
Machine View
Set Cursor Position
function number AH = 0x02 Set Cursor Position
register    value    explanation
BH          **      set videopage number
DH          Y        set cursor row
DL          X        set cursor col

Please push keyboard
keyboard 1 increment DH resister
keyboard 2 increment DL resister
keyboard 3 decrement DH resister
keyboard 4 decrement DL resister
keyboard esc return select mode
```

図 5.10 int10 キーボード 3 追加機能の実行結果(キーボード 3)

```
QEMU
Machine View
Set Cursor Position
function number AH = 0x02 Set Cursor Position
register    value    explanation
BH          **      set videopage number
DH          Y        set cursor row
DL          X        set cursor col

Please push keyboard
keyboard 1 increment DH resister
keyboard 2 increment DL resister
keyboard 3 decrement DH resister
keyboard 4 decrement DL resister
keyboard esc return select mode
```

図 5.11 int10 キーボード 3 追加機能の実行結果(キーボード 4)

5.4.4 Return Cursor Position

Return Cursor Position は、カーソル位置取得ファンクションを呼び出すことで、テキストモードのカーソル位置(X,Y)を取得する。X は列、Y は行でそれぞれ 0 から始まる。この命令は、選択画面からキーボード 4 を入力することによって実行可能になる。図 5.12 にキーボード 4 の初期画面を示す。追加機能は 5.4.3 の追加機能に加え、カーソル位置の取得を行う機能を追加している。カーソル位置は X が DL レジスタに、Y が DH レジスタに格納される。表 5.6 にキーボード 4 の追加機能の概要について示す。図 5.13 にカーソル位置の取得結果について示す。

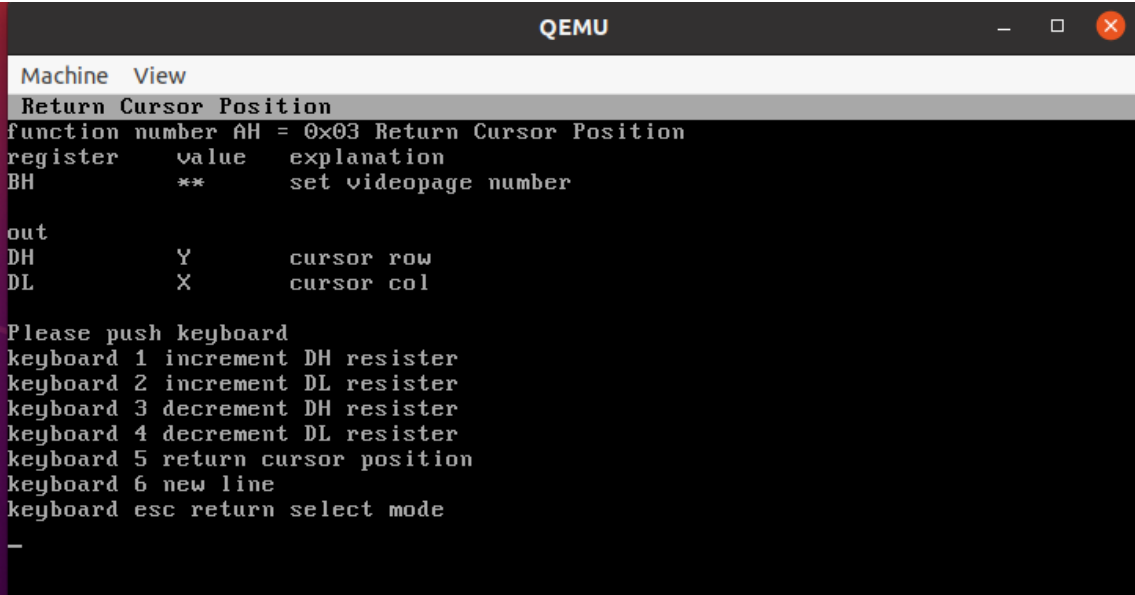


図 5.12 int10 キーボード 4 の初期画面

表 5.6 int10 キーボード 3 追加機能の概要

追加機能に割り当てたキーボード	詳細
1	一行下がる
2	一列右にずれる
3	一行上がる
4	一列左にずれる
5	カーソル位置の取得
6	改行

```
QEMU
Machine View
Return Cursor Position
function number AH = 0x03 Return Cursor Position
register    value    explanation
BH         **       set videopage number

out
DH          Y        cursor row
DL          X        cursor col

Please push keyboard
keyboard 1 increment DH resister
keyboard 2 increment DL resister
keyboard 3 decrement DH resister
keyboard 4 decrement DL resister
keyboard 5 return cursor position
keyboard 6 new line
keyboard esc return select mode

AX:0335 BX:0000 CX:0607 DX:1100 SI:A194 DI:0000
-
```

図 5.13 int10 キーボード 4 追加機能の実行結果(キーボード 5)

5.4.5 Return Light Pen Condition

Return Light Pen Condition は、ライトペン状態の取得である。ライトペンのステータス、位置などを取得する。この命令は通常の BIOS はこのファンクションが呼び出されても何もしないため、このアプリケーションによる命令の実行は不可能である。この命令はキーボード 5 に設定している。図 5.14 にキーボード 5 の画面を示す。

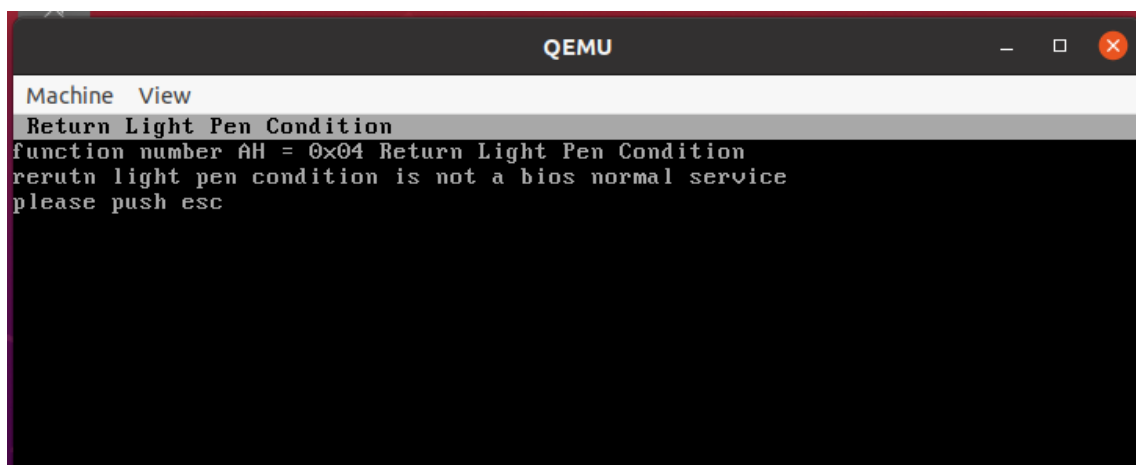


図 5.13 int10 キーボード 5 の実行結果

5.4.6 Set Current Video Page

Set Current Video Page はビデオページ設定である。ビデオバッファの現在選択中のページの変更を行える。ビデオバッファのアドレスは、モノクロは 0x000B0000、カラーは 0x000B8000、0x000A0000 である。この命令の追加機能は選択画面の遷移だけである。選択画面からキーボード 6 を入力することによって実行可能になる。図 5.14 にキーボード 6 の実行結果を示す。

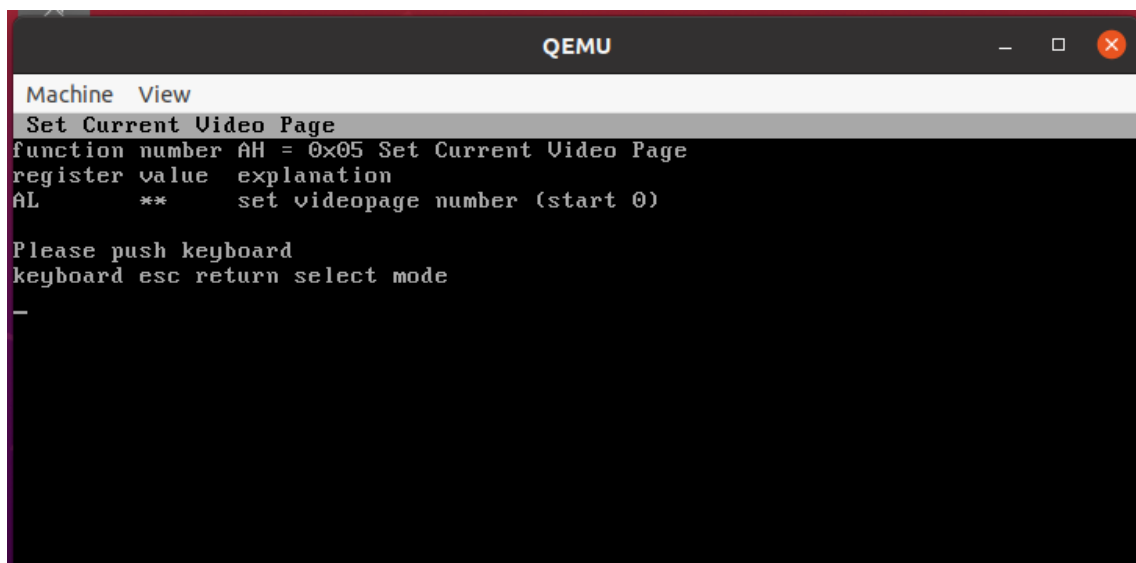


図 5.14 int10 キーボード 6 の実行結果

5.4.7 Scroll Up Region

Scroll Up Region は、上方向スクロールである。指定した行数分、上方向にスクロールする。アプリケーションによる実行は、選択画面から、キーボード 7 を入力することによって実行可能になる。図 5.15 にキーボード 7 の初期画面を示す。キーボード 7 は、追加機能として、一行スクロールを追加している。図 5.16 に上方向スクロールの実行結果を示す。

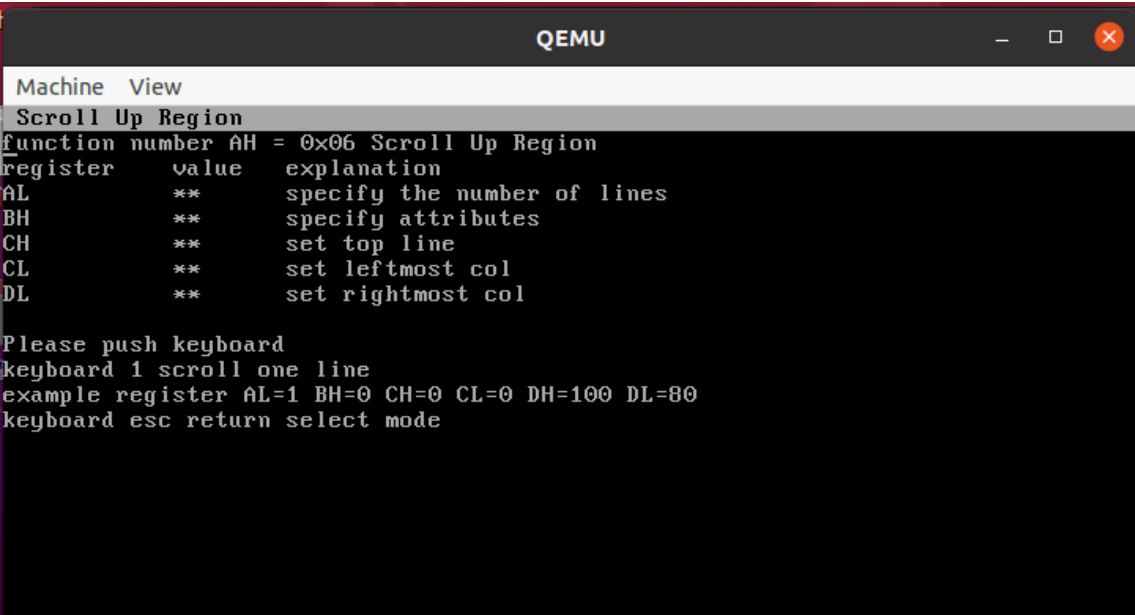


図 5.15 int10 キーボード 7 の初期画面

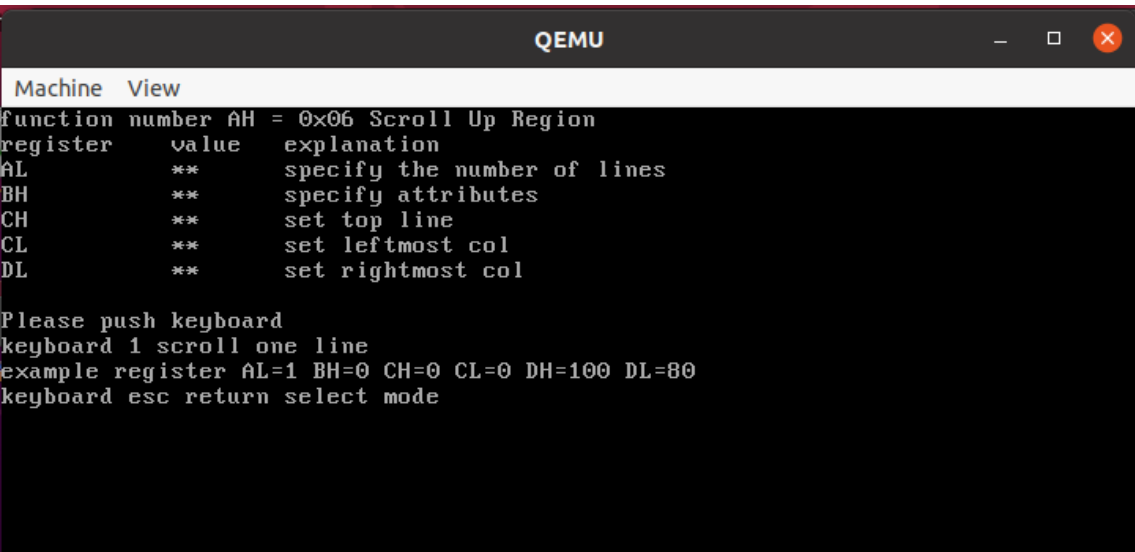


図 5.17 int10 キーボード 7 追加機能の実行結果(キーボード 1)

5.4.8 Scroll Down Region

Scroll Down Region は、下方向スクロールで、指定した行数分下方向スクロールを行う。
5.4.7 とは方向が違っただけで、レジスタに格納する値自体は同じである。アプリケーション
による実行は、選択画面から、キーボード 8 を入力することによって実行可能になる。図
5.18 にキーボード 8 の初期画面を、図 5.19 に下方向スクロールの実行結果を示す。

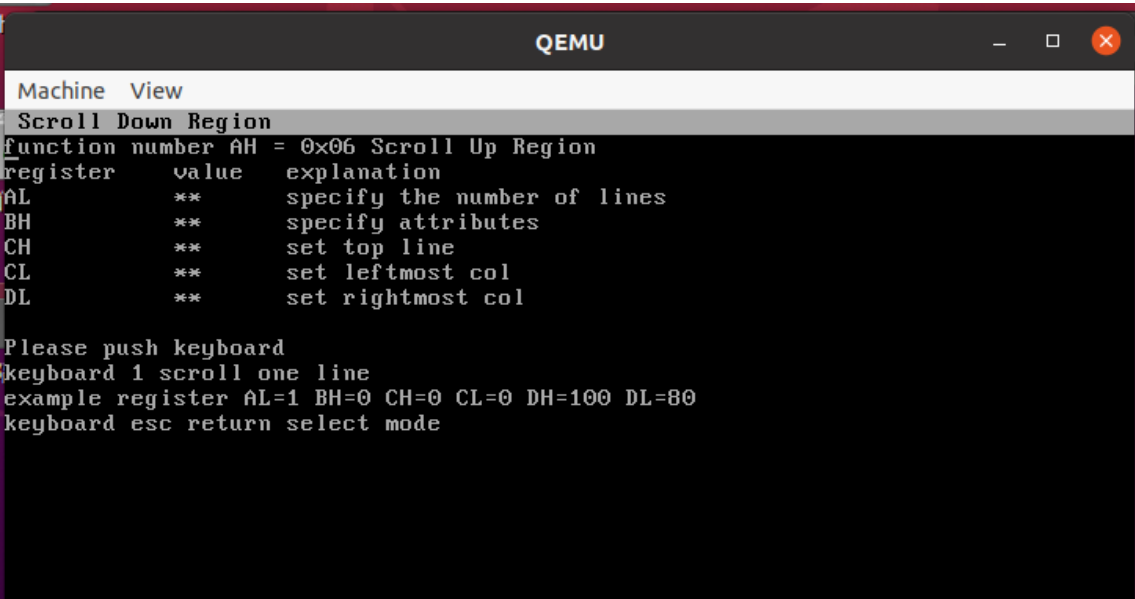


図 5.18 int10 キーボード 8 の初期画面

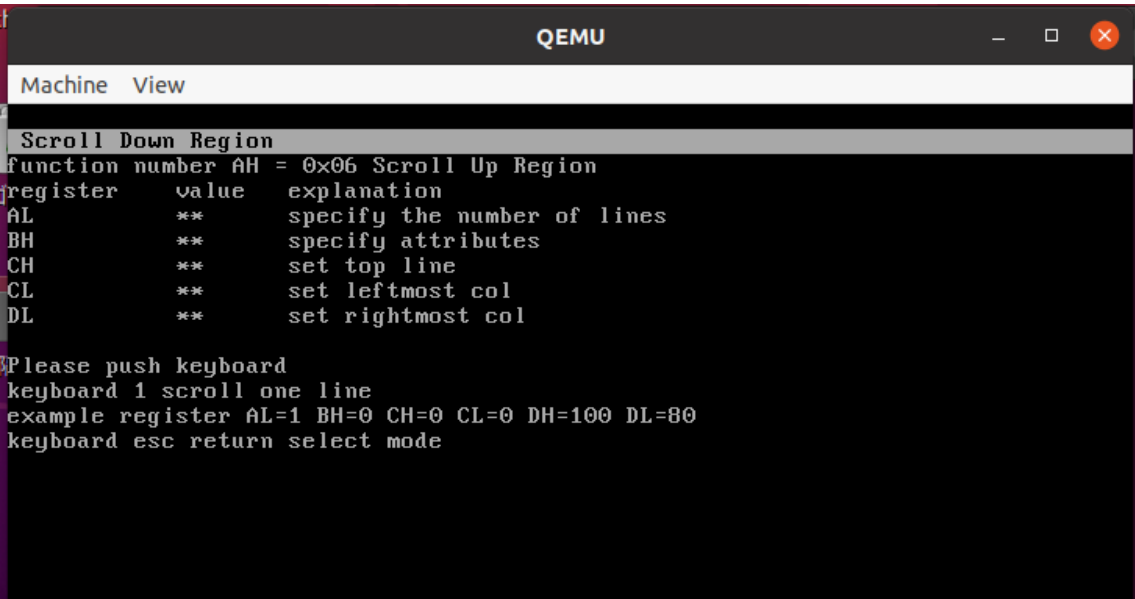


図 5.19 int10 キーボード 8 追加機能の実行結果(キーボード 1)

5.4.9 Return Character and Attribute

Return Character and Attribute は、文字コードとアトリビュート取得である。指定したページ上の、現在のカーソル上にある文字コードとアトリビュートを取得する。アトリビュートとは、属性のことであり、背景色などの情報を取得する。アプリケーションによる実行は、選択画面から、キーボード 9 を入力することによって実行可能になる。図 5.20 にキーボード 9 の初期画面を示す。追加機能は、文字コードとアトリビュート取得とレジスタの確認を行う。表 5.7 に追加機能の概要を示す。命令の実行を行った際、アトリビュートは AH レジスタに、文字コードは AL レジスタに格納される。図 5.21 に追加機能の実行結果を示す。

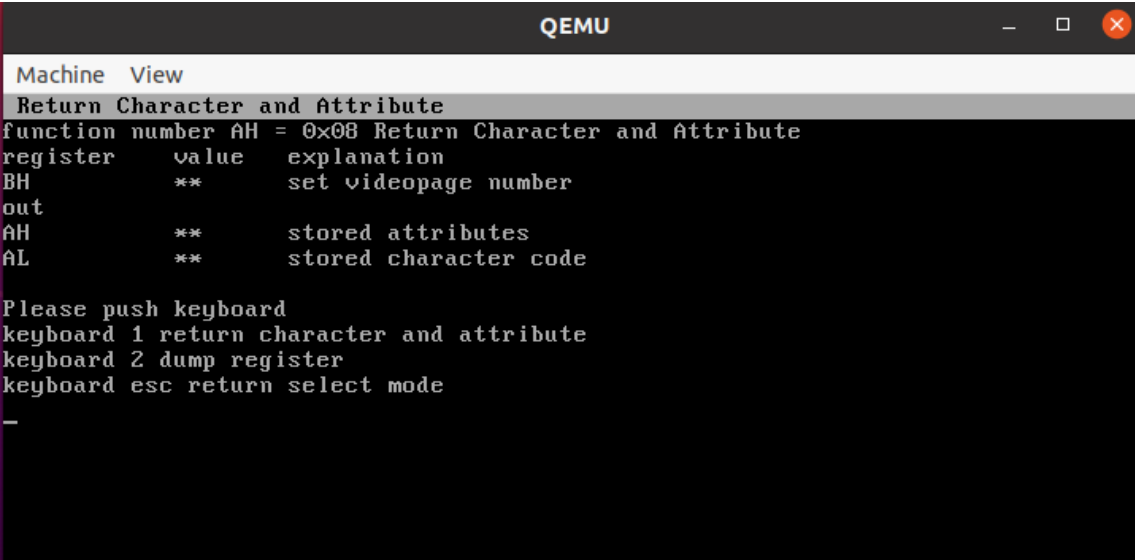


図 5.20 int10 キーボード 9 の初期画面

表 5.7 int10 キーボード 9 追加機能の概要

追加機能に割り当てたキーボード	詳細
1	文字コードとアトリビュート取得
2	レジスタのダンプ

```
Machine View
Return Character and Attribute
function number AH = 0x08 Return Character and Attribute
register    value  explanation
BH         **     set videopage number
out
AH         **     stored attributes
AL         **     stored character code

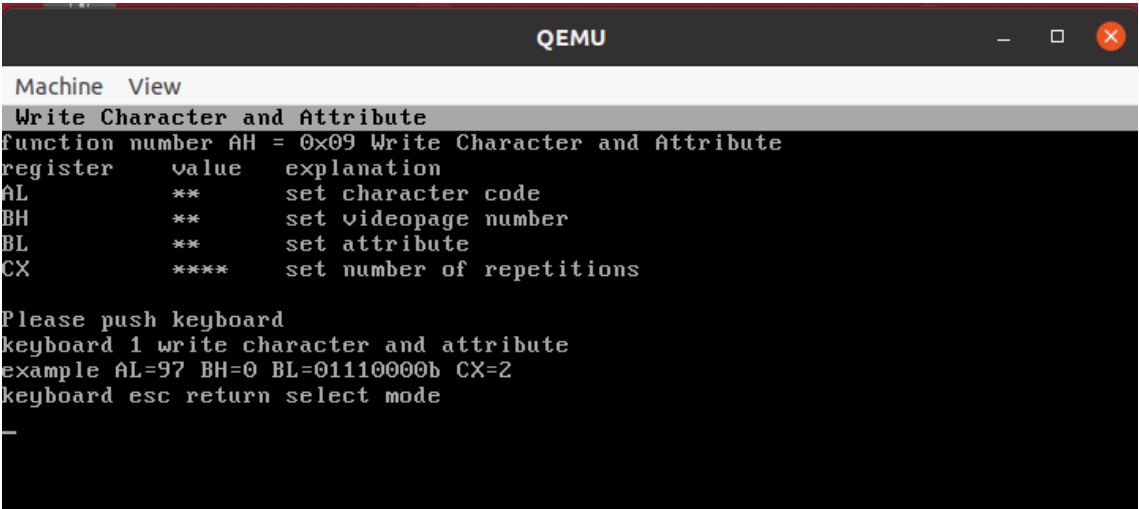
Please push keyboard
keyboard 1 return character and attribute
keyboard 2 dump register
keyboard esc return select mode

AX:0332 BX:0000 CX:0000 DX:0000 SI:0000 DI:0000
-
```

図 5.21 int10 キーボード 9 追加機能の実行結果(キーボード 1、2)

5.4.10 Write Character and Attribute

Write Character and Attribute は、文字コードとアトリビュート書き込みである。指定したページ上、現在のカーソル上に文字コードとアトリビュートを書き込む。繰り返し数を指定する項目があり、指定することで、列方向に繰り返し文字を書き込むことができる。アプリケーションによる実行は、選択画面から、キーボード a を入力することによって実行可能になる。図 5.22 にキーボード a の初期画面を示す。追加機能は、文字コードとアトリビュートの書き込みを行う。例は文字コードの格納をする AL レジスタに 97(ASCII コードで a)、ビデオページ番号を設定する BH レジスタに 0、アトリビュートを設定する BL レジスタに 0111000b を、繰り返し数である CX レジスタに 2 を格納している。図 5.23 に追加機能の実行結果を示す。



```
QEMU
Machine View
Write Character and Attribute
function number AH = 0x09 Write Character and Attribute
register    value    explanation
AL          **      set character code
BH          **      set videopage number
BL          **      set attribute
CX          ****     set number of repetitions

Please push keyboard
keyboard 1 write character and attribute
example AL=97 BH=0 BL=0111000b CX=2
keyboard esc return select mode
-
```

図 5.22 int10 キーボード a の初期画面

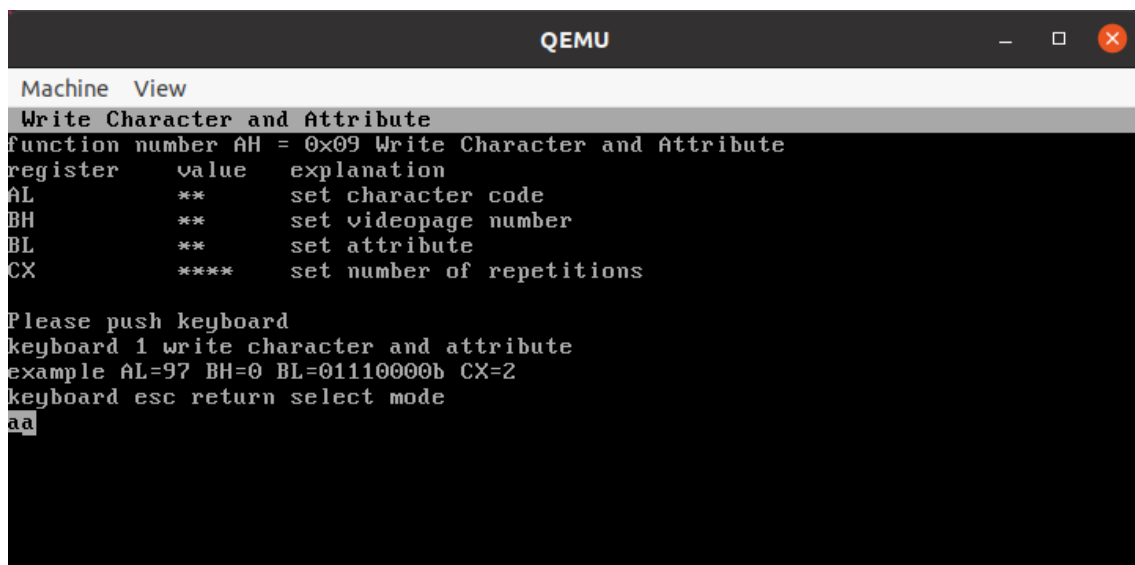
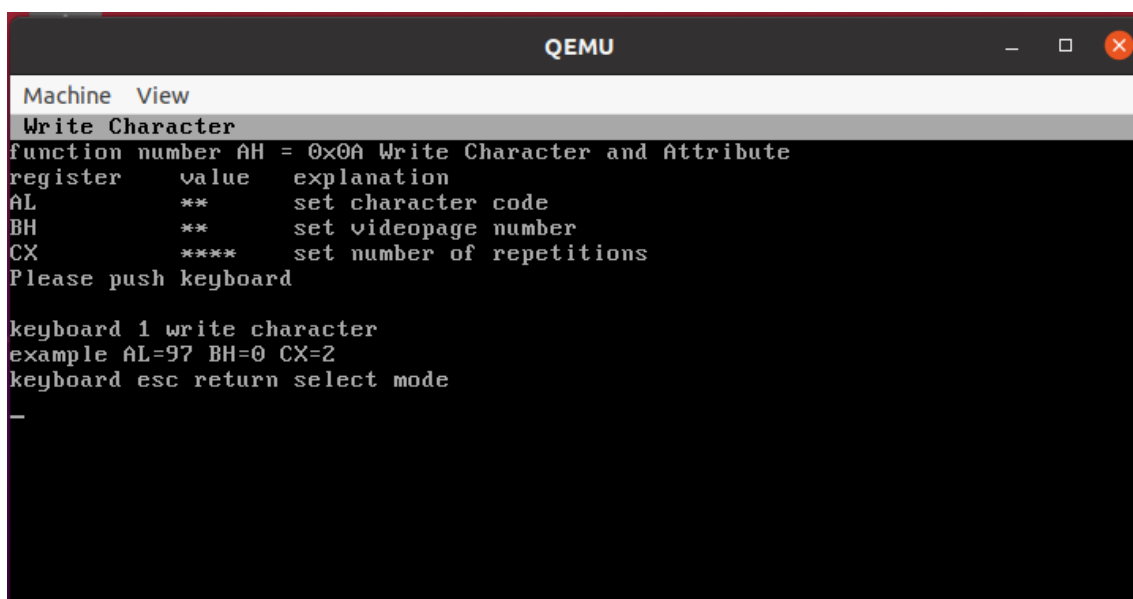


図 5.23 int10 キーボード a 追加機能の実行結果(キーボード 1)

5.4.11 Write Character

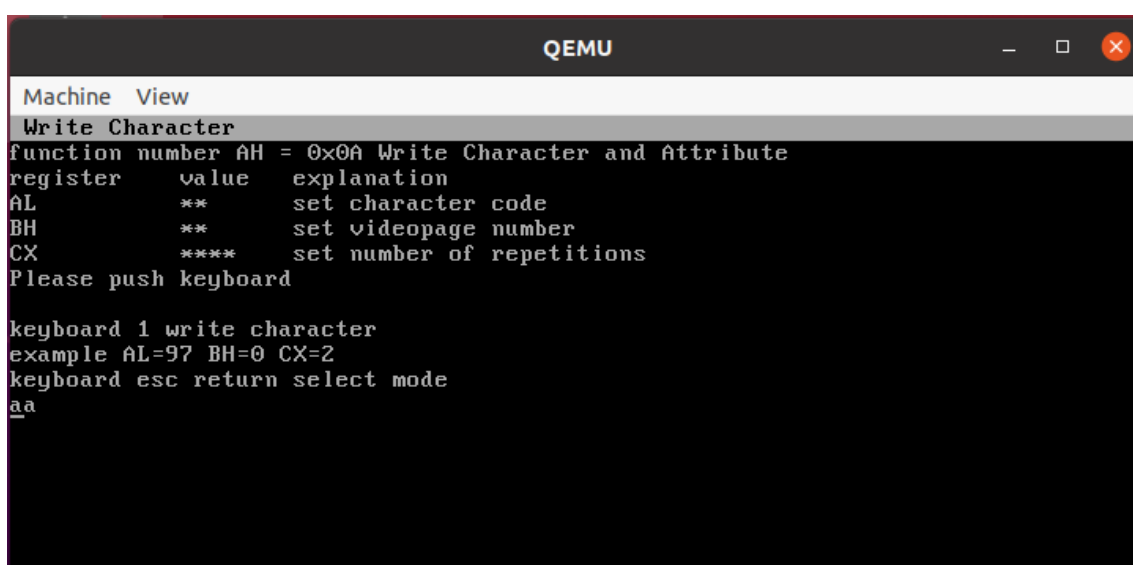
Write Character は文字コード書き込みである。5.4.10 の命令にアトリビュートの書き込みを省略し、文字コードのみの書き込みを行う命令である。アプリケーションによる実行は、選択画面から、キーボード b を入力することによって実行可能になる。図 5.24 にキーボード b の初期画面を示す。追加機能は、文字 a の出力を二回行う。図 5.25 に追加機能の実行結果を示す。



```
QEMU
Machine View
Write Character
function number AH = 0x0A Write Character and Attribute
register    value    explanation
AL          **      set character code
BH          **      set videopage number
CX          ****    set number of repetitions
Please push keyboard

keyboard 1 write character
example AL=97 BH=0 CX=2
keyboard esc return select mode
-
```

図 5.24 int10 キーボード b の初期画面



```
QEMU
Machine View
Write Character
function number AH = 0x0A Write Character and Attribute
register    value    explanation
AL          **      set character code
BH          **      set videopage number
CX          ****    set number of repetitions
Please push keyboard

keyboard 1 write character
example AL=97 BH=0 CX=2
keyboard esc return select mode
aa
```

図 5.25 int10 キーボード b 追加機能の実行結果(キーボード 1)

5.4.12 Set Color Palette

Set Color Palette は、カラーパレット設定である。グラフィックモード用のビデオパレットレジスタの値を設定する。追加機能は選択画面の遷移だけである。アプリケーションによる実行は、選択画面から、キーボード c を入力することによって実行可能になる。図 5.26 にキーボード c の初期画面を示す。

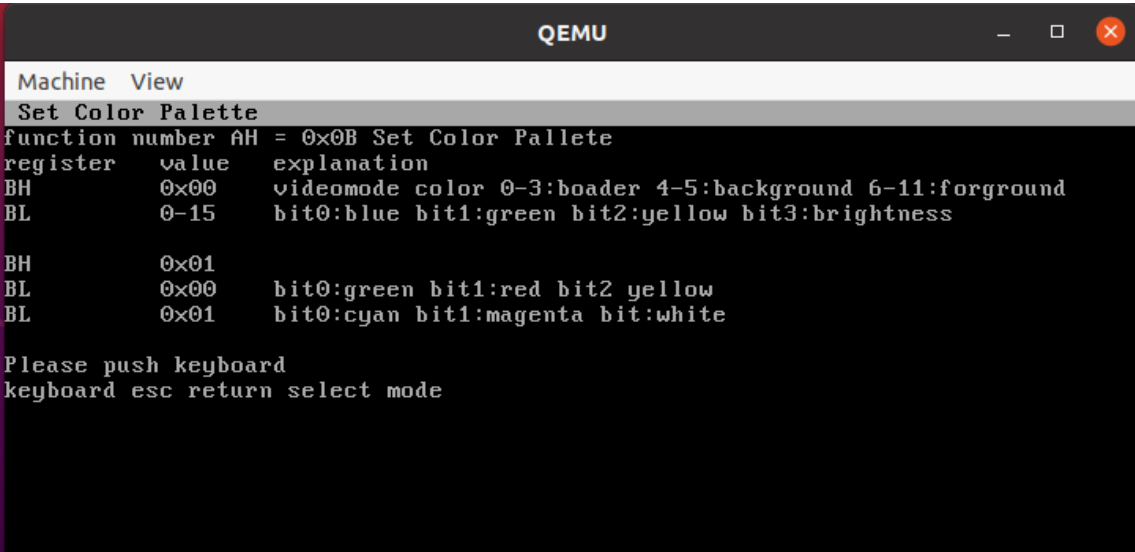


図 5.26 int10 キーボード c の初期画面

5.4.13 Write Graphic Pixel

Write Graphic Pixel は、ピクセル単位の書き込みである。指定した行と列のピクセルアドレスにカラーの値を書き込む。これは通常の bios サービスではないため、アプリケーションによる実行は不可能である。選択画面から d キーに割り当てている。図 5.27 に初期画面を示す。

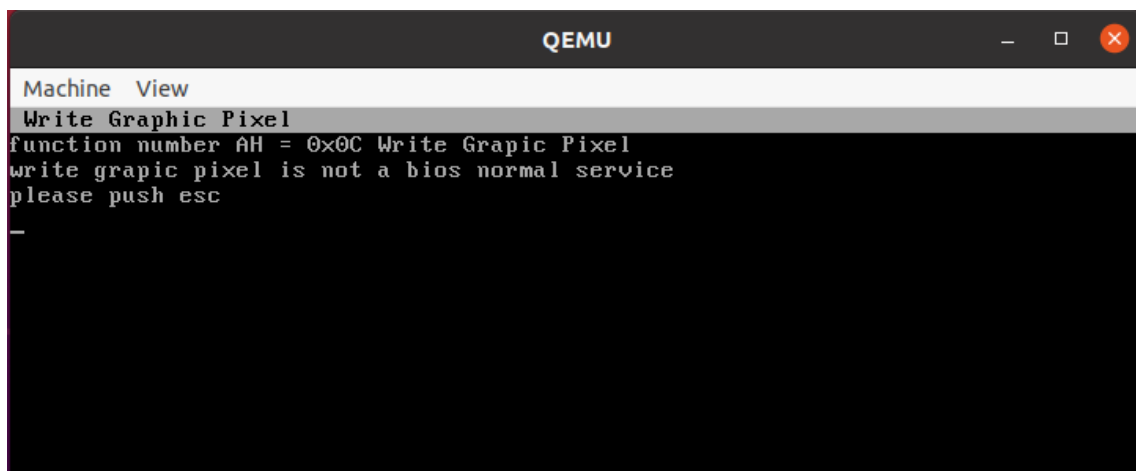


図 5.27 int10 キーボード d の初期画面

5.4.14 Read Graphic Pixel

Read Graphic Pixel は、ピクセル単位の書き込みである。指定した行と列のピクセルアドレスにカラーの値を読み込む。これは通常の bios サービスではないため、アプリケーションによる実行は不可能である。選択画面から e キーに割り当てている。図 5.28 に初期画面を示す。

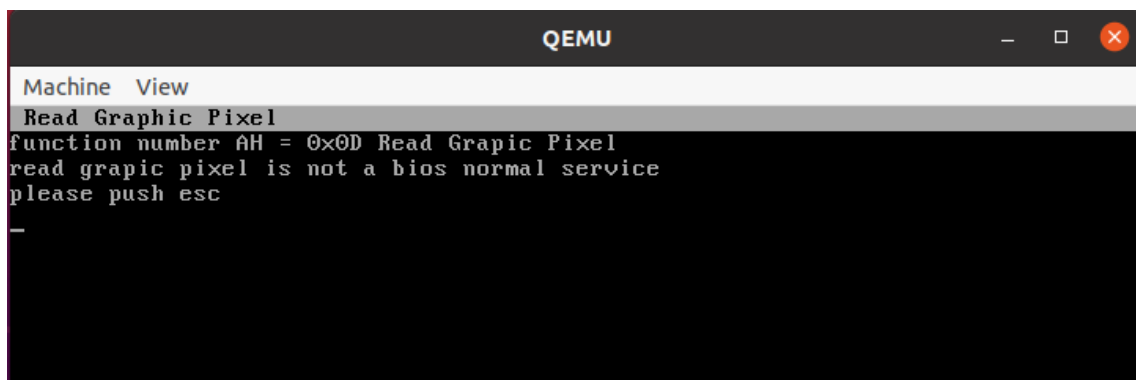
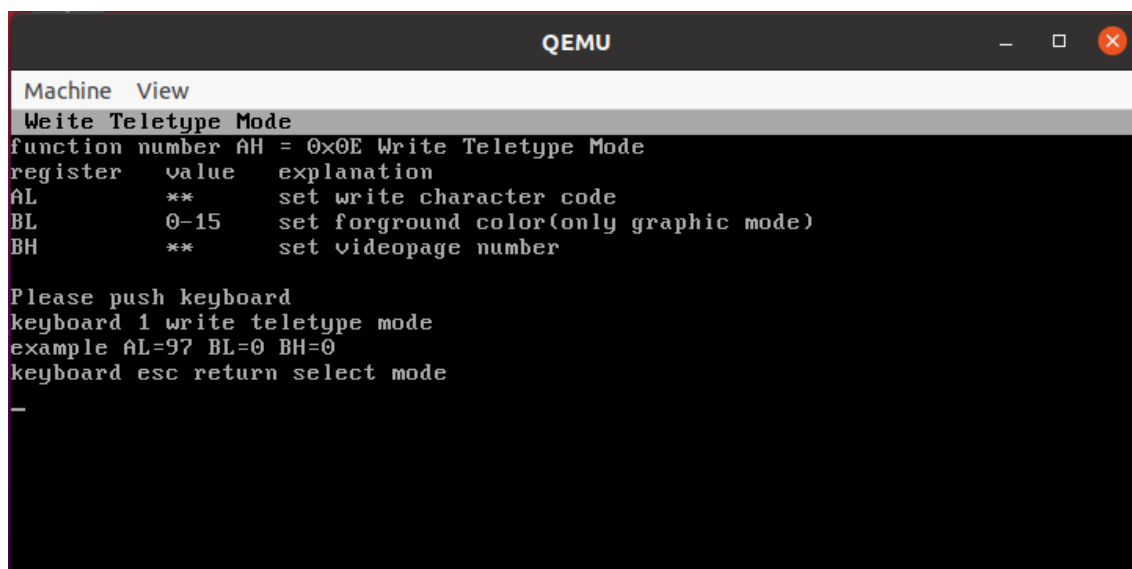


図 5.28 int10 キーボード e の初期画面

5.4.15 Write Teletype Mode

Write Teletype Mode はテレタイプ出力による文字コード書き込みである。現在のカーソル上に指定する文字コードの書き込みを行う。この命令後、カーソルが次の行に移動する。カーソルが一番右端に存在する場合は、列を 0 にリセットし、次の行に移動する。現在のカーソルがスクリーンの一番右端で、一番下にあった場合は画面をスクロールする。アプリケーションによる実行は、選択画面から、キーボード f を入力することによって実行可能になる。図 5.29 にキーボード f の初期画面を示す。追加機能は小文字 a を出力する機能を追加した。図 5.30 に追加機能の実行結果を示す。

The image shows a QEMU terminal window with a dark background and light-colored text. The window title is 'QEMU'. Inside the terminal, the text 'Machine View' is at the top. Below it, 'Write Teletype Mode' is displayed in a larger font. The main content is a list of registers and their functions: 'function number AH = 0x0E Write Teletype Mode', 'register value explanation', 'AL ** set write character code', 'BL 0-15 set foreground color(only graphic mode)', and 'BH ** set videopage number'. Below this, there are instructions: 'Please push keyboard', 'keyboard 1 write teletype mode', 'example AL=97 BL=0 BH=0', and 'keyboard esc return select mode'. A single hyphen '-' is at the bottom of the terminal window.

```
Machine View
Write Teletype Mode
function number AH = 0x0E Write Teletype Mode
register value explanation
AL ** set write character code
BL 0-15 set foreground color(only graphic mode)
BH ** set videopage number

Please push keyboard
keyboard 1 write teletype mode
example AL=97 BL=0 BH=0
keyboard esc return select mode
-
```

図 5.29 int10 キーボード f の初期画面

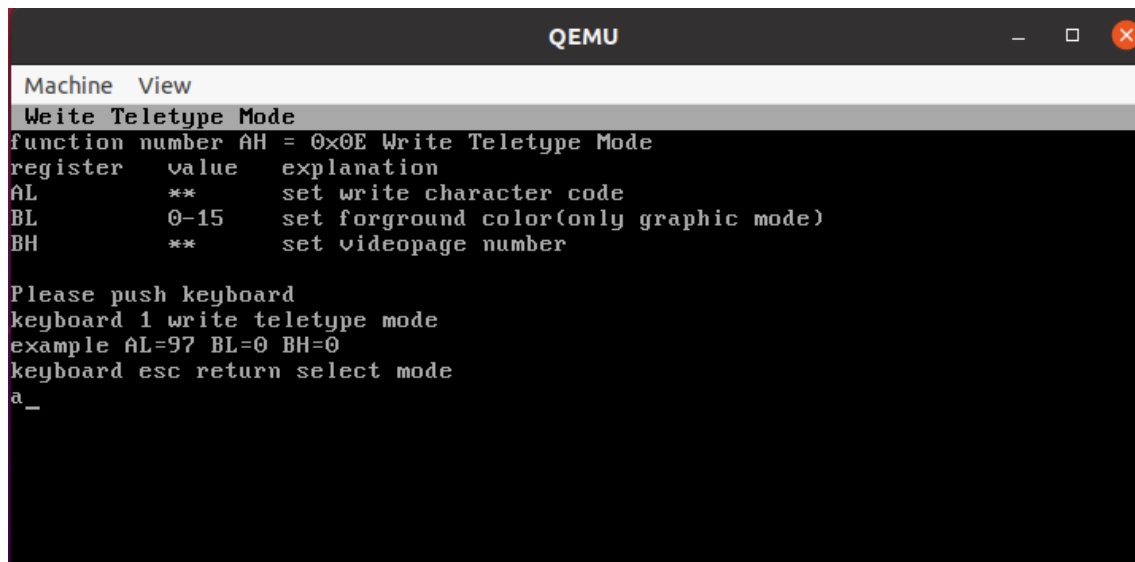


図 5.30 int10 キーボード f 追加機能の実行結果(キーボード 1)

5.4.16 Return Video Display Mode

Return Video Display Mode は、ディスプレイモードの取得である。スクリーン上の列数と現在のビデオモードを取得する。アプリケーションによる実行は、選択画面から、キーボード g を入力することによって実行可能になる。図 5.31 にキーボード g の初期画面を示す。追加機能は、ディスプレイモードの取得とともに、レジスタに格納した値を出力する。AH レジスタはスクリーン上の列数、AL レジスタは現在のビデオモード、BH レジスタは現在のビデオページ番号が格納される。図 5.32 に追加機能の実行結果を示す。

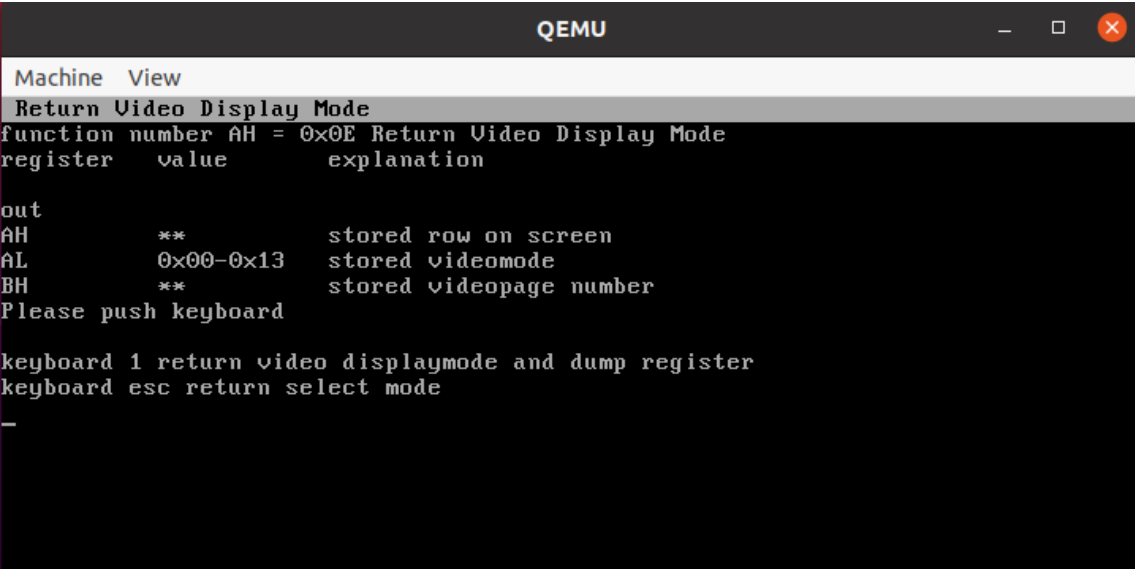


図 5.31 int10 キーボード g の初期画面

```
QEMU
Machine View
Return Video Display Mode
function number AH = 0x0E Return Video Display Mode
register    value    explanation
out
AH          **      stored row on screen
AL          0x00-0x13 stored videomode
BH          **      stored videopage number
Please push keyboard

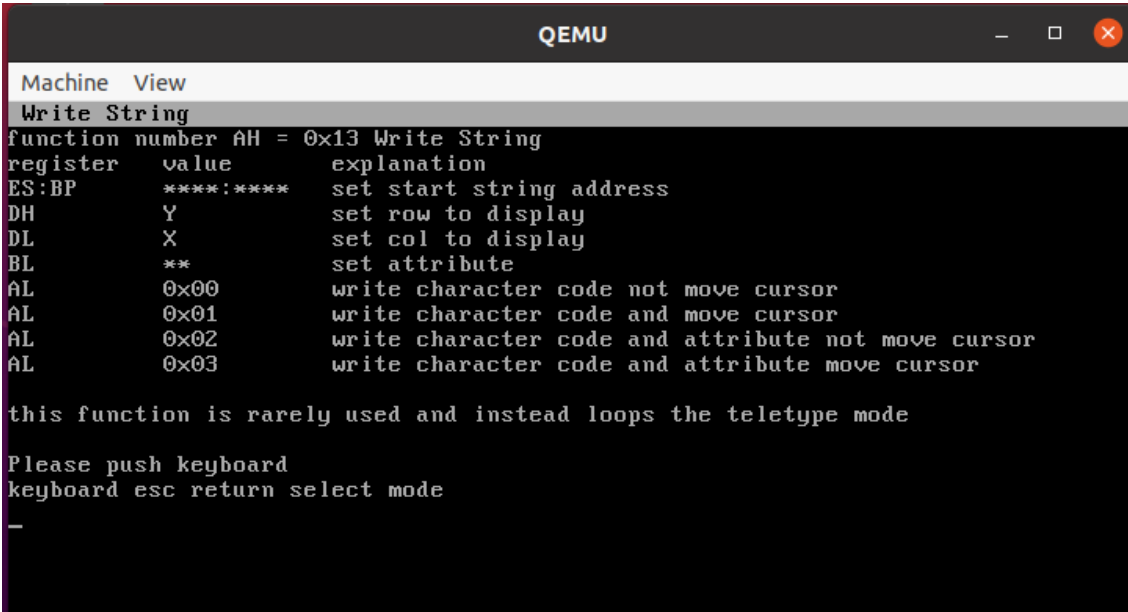
keyboard 1 return video displaymode and dump register
keyboard esc return select mode

AX:5003 BX:0000 CX:0002 DX:0000 SI:0000 DI:0000
-
```

図 5.32 int10 キーボード g 追加機能の実行結果(キーボード 1)

5.4.17 Write String

Write String は文字列書き込みである。指定したアドレス上の文字列を書き込む。しかしながら、この命令は滅多に使用されることは無く、代わりにテレタイプ出力をループさせることで文字列の書き込みを行う方法が使用される。そのため、追加機能は選択画面の遷移のみである。選択画面に割り当てたキーボードは、h に割り当てている。図 5.32 に初期画面を示す。



The screenshot shows a QEMU window titled "QEMU" with a "Machine View" tab selected. The "Write String" function is displayed, showing its details and a list of registers and their values. The registers listed are ES:BP, DH, DL, BL, AL, and AL, with their respective values and explanations. The text "this function is rarely used and instead loops the teletype mode" is also visible, along with instructions to "Please push keyboard" and "keyboard esc return select mode".

```
Machine View
Write String
function number AH = 0x13 Write String
register    value    explanation
ES:BP      ****:****  set start string address
DH         Y        set row to display
DL         X        set col to display
BL         **        set attribute
AL         0x00      write character code not move cursor
AL         0x01      write character code and move cursor
AL         0x02      write character code and attribute not move cursor
AL         0x03      write character code and attribute move cursor

this function is rarely used and instead loops the teletype mode

Please push keyboard
keyboard esc return select mode
-
```

図 5.32 int10 キーボード h の初期画面

5.5 bios コール 0x16

bios コール 0x16 ビデオサービスの概要を表 5.8 に示す。表 5.8 は 0x16 によって実行可能なサービスの種類である。

表 5.8 0x16 ビデオサービスの概要

機能番号	ファンクション名	概要
0x00	Read Keyboard Input	キーボード入力読み込み
0x01	Return Keyboard Status	キーボードステータスの取得
0x02	Return Shift Flag Status	シフトフラグステータスの取得
0x03	Set Typematic Rate	キーボード、レートの設定
0x05	Push Data to Keyboard	キーデータの書き込み
0x10	Enhanced Read Keyboard	拡張キーボードの入力読み込み
0x11	Enhanced Read Keyboard Status	拡張キーボードのステータス読み込み
0x12	Enhanced Read Keyboard Flags	拡張キーボードのフラグ読み込み
0xF0	Set CPU Speed	CPU のクロック設定
0xF1	Get CPU Speed	CPU クロック設定を取得
0xF400	Read Cache Status	キャッシュステータスの読み込み
0xF401	Enable Cache	キャッシュの有効化
0xF402	Disable Cache	キャッシュの無効化

5.6 アプリケーション int16.asm について

アプリケーションは、表 5.8 のビデオサービスを実行可能である。表 5.9 に割り当てたキーボードの詳細を示す。

表 5.9 0x16 ビデオサービスの概要

ファンクション名	割り当てたキーボード
Read Keyboard Input	1
Return Keyboard Status	2
Return Shift Flag Status	3
Set Typematic Rate	4
Push Data to Keyboard	5
Enhanced Read Keyboard	6
Enhanced Read Keyboard Status	7
Enhanced Read Keyboard Flags	8
Set CPU Speed	9
Get CPU Speed	a
Read Cache Status	b
Enable Cache	c
Disable Cache	d

5.6.1 Read Keyboard Input

Read Keyboard Input は、キーボード入力読み込みである。キーボード入力を読み込むことができ、命令の実行時、キーボードが入力されるまで待ち続ける。キーボード入力時に、AH レジスタにスキャンコードが格納され、AL レジスタに ASCII コードが格納される。特殊例として、ALT キーがある。ALT キーが押された場合アスキーコードに 0x00 が格納される。アプリケーションによる実行は、選択画面から、キーボード 1 を入力することによって実行可能になる。図 5.6.1 にキーボード 1 の初期画面を示す。追加機能は、キーボード入力後レジスタの値を表示する機能を追加した。追加機能はキーボード 1 入力後、キーボードが入力されるまで待機する。その後任意のキーボードを入力すれば、レジスタの値がダンプされる。図 5.6.2 に追加機能の実行結果を示す。これは、キーボード a を入力した場合である。

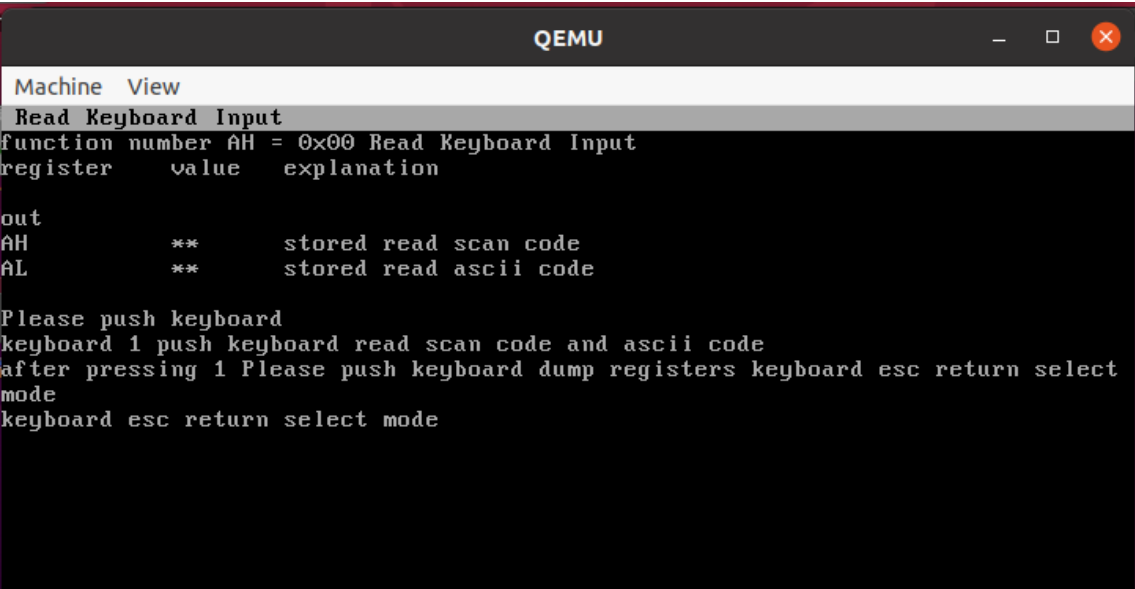


図 5.6.1 int16 キーボード 1 の初期画面

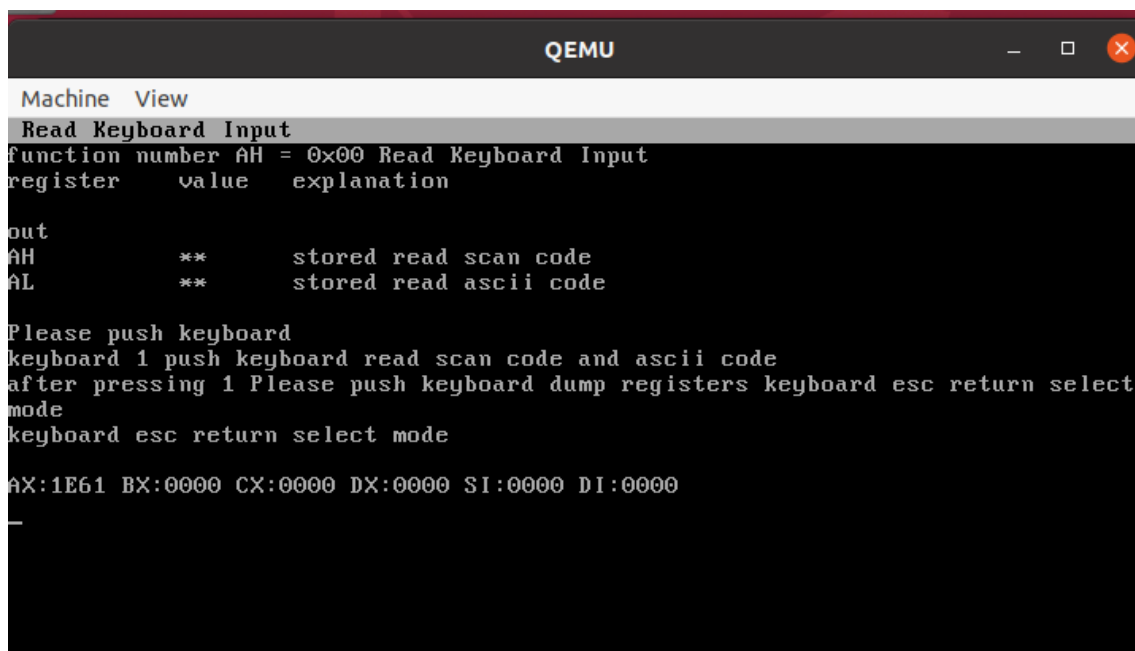
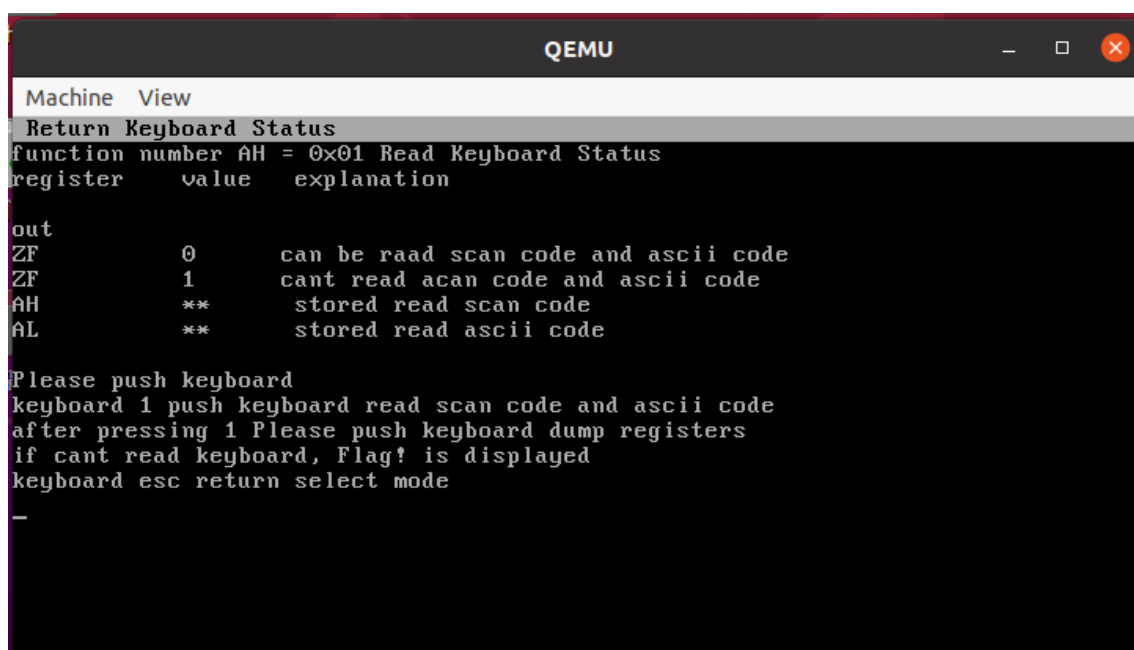


図 5.6.2 int16 キーボード 1 追加機能の実行結果(キーボード 1 a 入力)

5.6.2 Return Keyboard Status

Return Keyboard Status は、キーボードステータス読み込みである。キーボード入力読み込みで読み込みが可能であるかを判断できる。データが無い場合、EFLAGS の ZF フラグが 0 にクリアされる。読み込めるデータがある場合、EFLAGS の ZF フラグに 1 がセットされ、AH レジスタにスキャンコードが、AL レジスタに ASCII コードが格納される。この命令はキーボードの入力が読み込めるが、キーボードのデータ入力レジスタはクリアされない。キーボード入力読み込みでのみクリアされる。アプリケーションによる実行は、選択画面から、キーボード 2 を入力することによって実行可能になる。図 5.6.3 にキーボード 2 の初期画面を示す。追加機能は、5.6.1 の命令にエラーチェックを加えた機能である。読み取れないデータならば、ゼロフラグにチェックが入る。図 5.6.4 に追加機能の実行結果を示す。



```
QEMU
Machine View
Return Keyboard Status
function number AH = 0x01 Read Keyboard Status
register    value  explanation
out
ZF          0      can be read scan code and ascii code
ZF          1      cant read scan code and ascii code
AH          **     stored read scan code
AL          **     stored read ascii code

Please push keyboard
keyboard 1 push keyboard read scan code and ascii code
after pressing 1 Please push keyboard dump registers
if cant read keyboard, Flag! is displayed
keyboard esc return select mode
-
```

図 5.6.3 int16 キーボード 2 の初期画面

```
QEMU
Machine View
Return Keyboard Status
function number AH = 0x01 Read Keyboard Status
register    value    explanation
out
ZF          0        can be raad scan code and ascii code
ZF          1        cant read acan code and ascii code
AH          **       stored read scan code
AL          **       stored read ascii code

Please push keyboard
keyboard 1 push keyboard read scan code and ascii code
after pressing 1 Please push keyboard dump registers
if cant read keyboard, Flag! is displayed
keyboard esc return select mode

AX:0161 BX:0000 CX:0000 DX:0000 SI:0000 DI:0000
-
```

図 5.6.4 int16 キーボード 2 追加機能の実行結果(キーボード 1 a 入力)

5.6.3 Return Shift Flag Status

Return Shift Flag Status は、シフトフラグステータス取得である。シフトキーなどのキーのステータスを取得する。表 5.6.2 にキーのステータスの概要を示す。アプリケーションによる実行は、選択画面から、キーボード 3 を入力することによって実行可能になる。追加機能は、選択画面の遷移だけである。図 5.6.5 にキーボード 3 の実行画面を示す。

表 5.6.2 キーのステータス概要

AL レジスタに出力される値	概要
10000000b	インサートキーがアクティブ
01000000b	キャプスロックキーがアクティブ
00010000b	ナムロックキーがアクティブ
00001000b	ALT キーが押されている状態
00000100b	CTRL キーが押されている状態
00000010b	左シフトキーが押されている状態
00000001b	右シフトキーが押されている状態

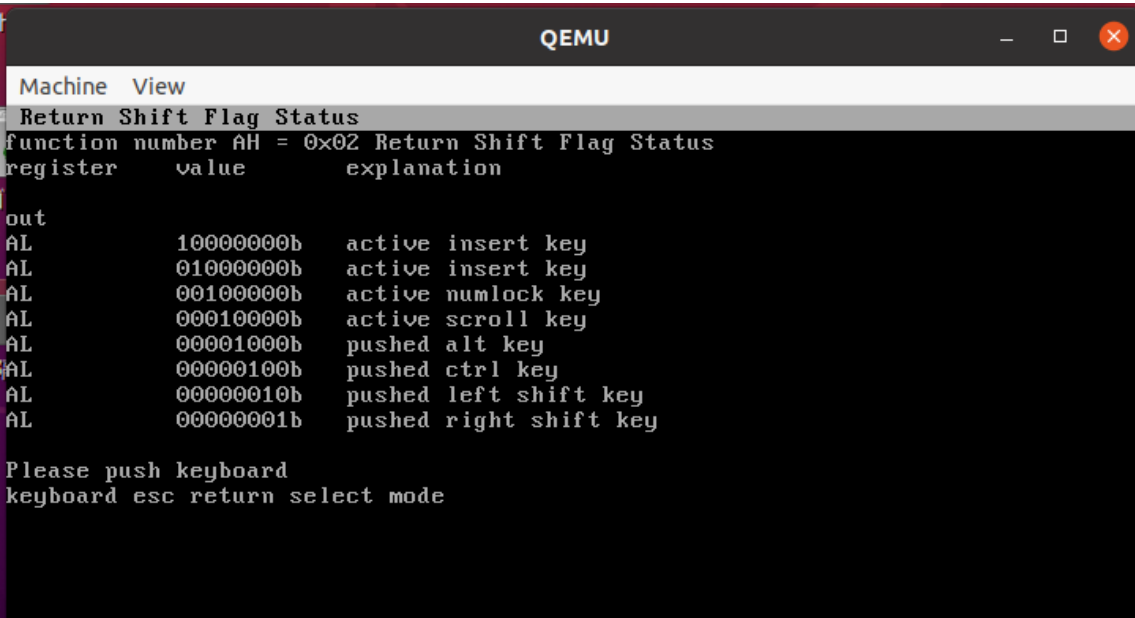
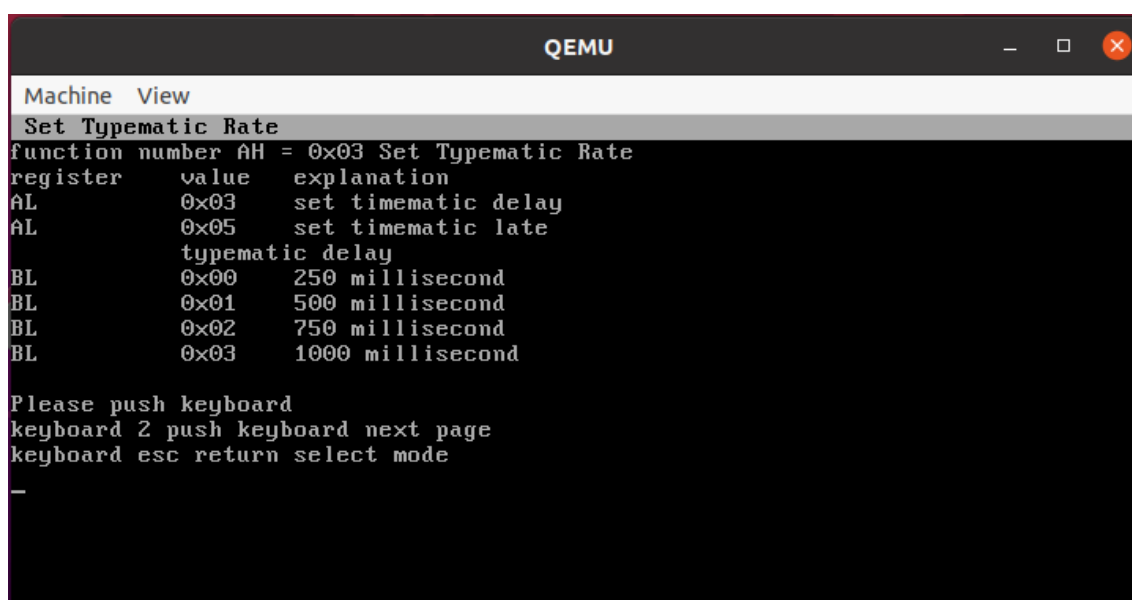


図 5.6.5 int16 キーボード 3 の実行画面

5.6.4 Set Typematic Rate

Set Typematic Rate はタイプマティック設定である。タイプマティック遅延というキーが押されている状態で、連続で押下する状態を検討する間隔と、一秒間に何個のキーを検出するかを決めるタイプマティックレートの設定を行う。アプリケーションによる実行は、選択画面から、キーボード 4 を入力することによって実行可能になる。追加機能には、命令の実行は無く、一画面にテキストが収まらないため、ページごとに区切っている。図 5.6.6 にキーボード 4 のページ 1 を、図 5.6.7 にページ 2 を、図 5.6.8 にページ 3 を示す。



```
QEMU
Machine View
Set Typematic Rate
function number AH = 0x03 Set Typematic Rate
register    value    explanation
AL          0x03     set timematic delay
AL          0x05     set timematic late
              typematic delay
BL          0x00     250 millisecond
BL          0x01     500 millisecond
BL          0x02     750 millisecond
BL          0x03     1000 millisecond

Please push keyboard
keyboard 2 push keyboard next page
keyboard esc return select mode
-
```

図 5.6.6 int16 キーボード 4 の実行画面(ページ 1)

Machine View		
register	value	explanation
		typematic latemematic delay
BL	0x00	30.0 characters per sec
BL	0x01	26.7 characters per sec
BL	0x02	24.0 characters per sec
BL	0x03	21.8 characters per sec
BL	0x04	20.0 characters per sec
BL	0x05	18.5 characters per sec
BL	0x06	27.1 characters per sec
BL	0x07	16.0 characters per sec
BL	0x08	23.1 characters per sec
BL	0x09	13.3 characters per sec
BL	0x0A	12.0 characters per sec
BL	0x0B	10.9 characters per sec
BL	0x0C	10.0 characters per sec
BL	0x0D	9.2 characters per sec
BL	0x0E	8.6 characters per sec
BL	0x0F	8.0 characters per sec
BL	0x10	7.5 characters per sec
BL	0x11	6.7 characters per sec

Please push keyboard
 keyboard 3 push keyboard next page
 keyboard esc return select mode

図 5.6.7 int16 キーボード 4 の実行画面(ページ 2)

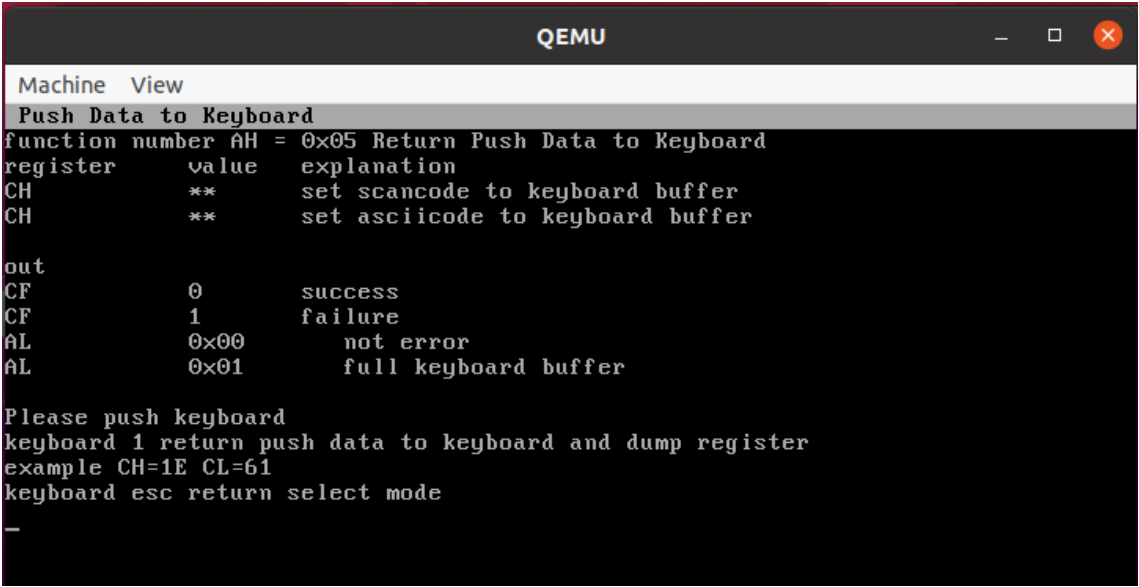
Machine View		
BL	0x12	6.0 characters per secc
BL	0x13	5.5 characters per secc
BL	0x14	5.0 characters per secc
BL	0x15	4.6 characters per sec
BL	0x16	4.3 characters per sec
BL	0x17	4.0 characters per sec
BL	0x18	3.7 characters per sec
BL	0x19	3.3 characters per sec
BL	0x1A	3.1 characters per sec
BL	0x1B	2.7 characters per sec
BL	0x1C	2.5 characters per sec
BL	0x1D	2.3 characters per sec
BL	0x1E	2.1 characters per sec
BL	0x1F	2.0 characters per sec

Please push keyboard
 keyboard 1 push keyboard return to the first page
 keyboard esc return select mode

図 5.6.7 int16 キーボード 4 の実行画面(ページ 3)

5.6.5 Push Data to Keyboard

Push Data to Keyboard は、キーボードバッファデータ書き込みである。キーボードのレジスタにスキャンコードとアスキーコードを書き込む命令である。入力には、CH レジスタにスキャンコードを、CL レジスタにアスキーコードを指定する。アプリケーションによる実行は、選択画面から、キーボード 5 を入力することによって実行可能になる。図 5.6.8 にキーボード 5 の初期画面を示す。追加機能は命令の実行と共に、レジスタの値を表示する。例として CH レジスタに 1E を、CL レジスタに 61 を格納している。図 5.6.9 に追加機能の実行結果を示す。



```
QEMU
Machine View
Push Data to Keyboard
function number AH = 0x05 Return Push Data to Keyboard
register    value  explanation
CH         **    set scancode to keyboard buffer
CH         **    set ascii-code to keyboard buffer

out
CF         0      success
CF         1      failure
AL         0x00    not error
AL         0x01    full keyboard buffer

Please push keyboard
keyboard 1 return push data to keyboard and dump register
example CH=1E CL=61
keyboard esc return select mode
-
```

図 5.6.8 int16 キーボード 5 の初期画面

```
QEMU
Machine View
Push Data to Keyboard
function number AH = 0x05 Return Push Data to Keyboard
register    value    explanation
CH          **       set scancode to keyboard buffer
CH          **       set asciiicode to keyboard buffer

out
CF          0        success
CF          1        failure
AL          0x00      not error
AL          0x01      full keyboard buffer

Please push keyboard
keyboard 1 return push data to keyboard and dump register
example CH=1E CL=61
keyboard esc return select mode

AX:0500 BX:0000 CX:1E61 DX:0000 SI:0000 DI:0000
-
```

図 5.6.9 int16 キーボード 5 追加機能の実行結果(キーボード 1)

5.6.6 Enhanced Read Keyboard

Enhanced Read Keyboard は、拡張キーボード読み込みファンクションである。101 キーボードの入力を読み込むことが可能で、動作は 5.6.1 の Read Keyboard Input と同じである。このファンクションの互換機能が DOS に実装されているため、通常ではこのファンクションは使用しない。アプリケーションによる実行は、選択画面から、キーボード 6 を入力することによって実行可能になる。図 5.6.10 にキーボード 6 の初期画面を示す。追加機能の動作は 5.6.1 と同じである。図 5.6.11 に追加機能の実行結果を示す。

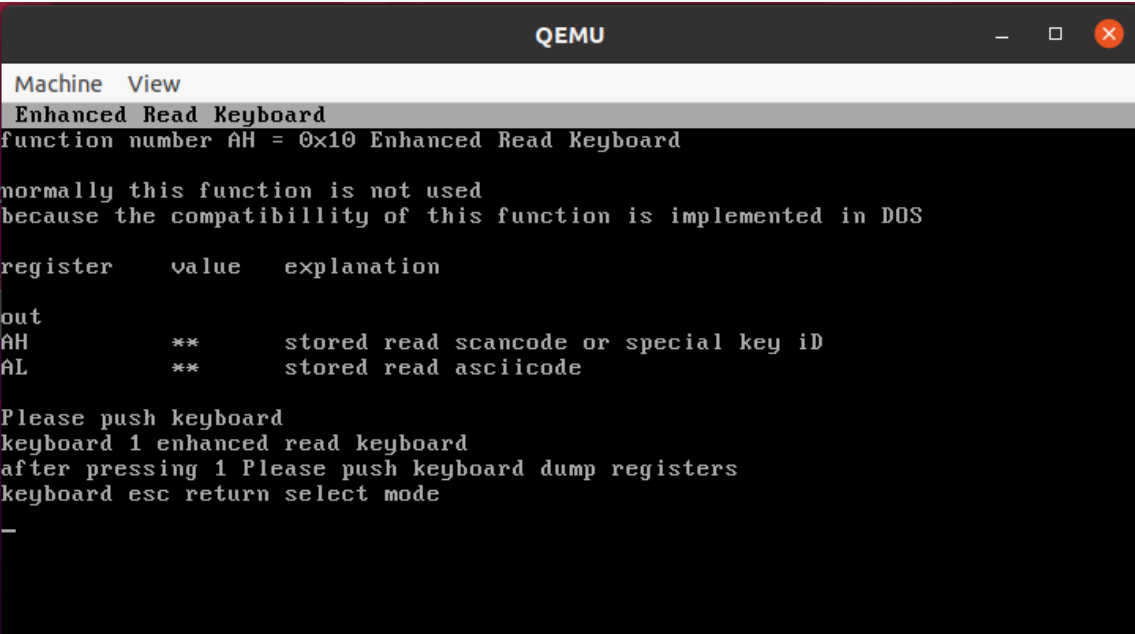
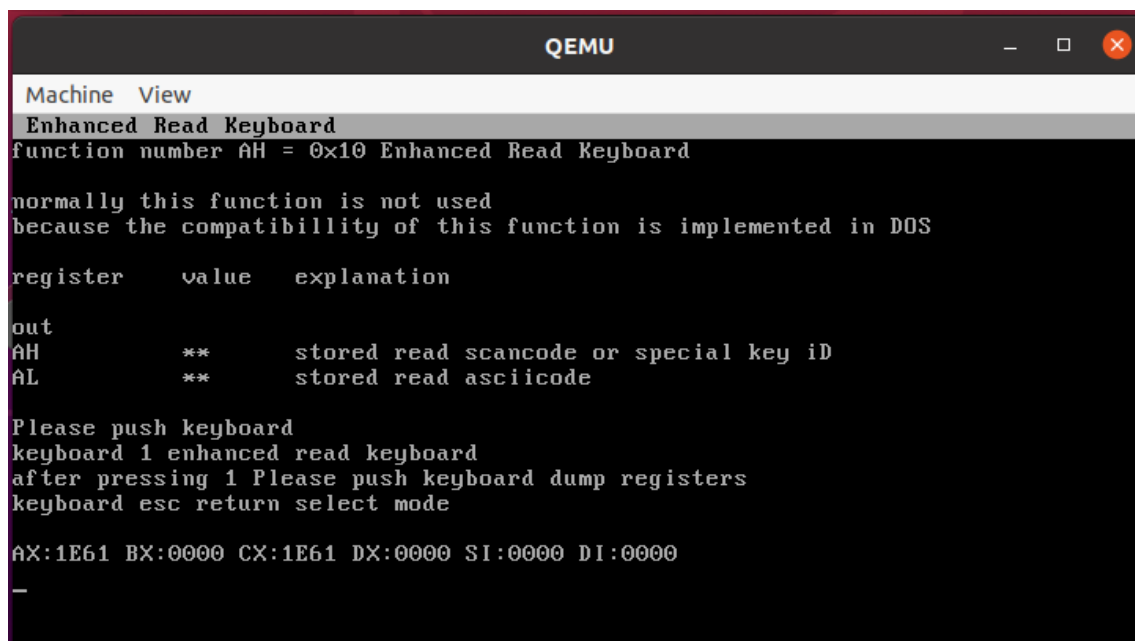


図 5.6.10 int16 キーボード 6 の初期画面



```
Machine View
Enhanced Read Keyboard
function number AH = 0x10 Enhanced Read Keyboard

normally this function is not used
because the compatibillity of this function is implemented in DOS

register    value    explanation
out
AH          **      stored read scancode or special key iD
AL          **      stored read asciicode

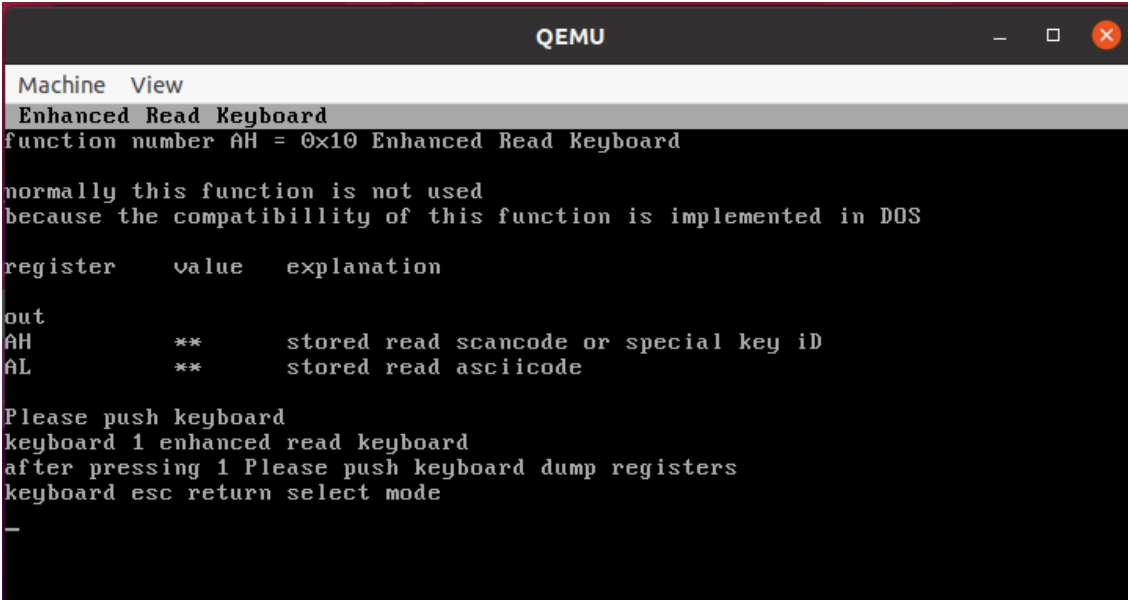
Please push keyboard
keyboard 1 enhanced read keyboard
after pressing 1 Please push keyboard dump registers
keyboard esc return select mode

AX:1E61 BX:0000 CX:1E61 DX:0000 SI:0000 DI:0000
-
```

図 5.6.9 int16 キーボード 6 追加機能の実行結果(キーボード 1 a 入力)

5.6.7 Enhanced Read Keyboard Status

Enhanced Read Keyboard Status は拡張キーボード読み込みファンクションである。101 キーのステータスを読み込む。動作は 5.6.2 の Return Keyboard Status と同じである。このファンクションの互換機能が DOS に実装されているため、通常では、このファンクションは使用しない。アプリケーションによる実行は、選択画面から、キーボード 7 を入力することによって実行可能になる。図 5.6.12 にキーボード 7 の初期画面を示す。追加機能の動作は 5.6.2 の動作と同じである。図 5.6.13 に追加機能の実行結果を示す。



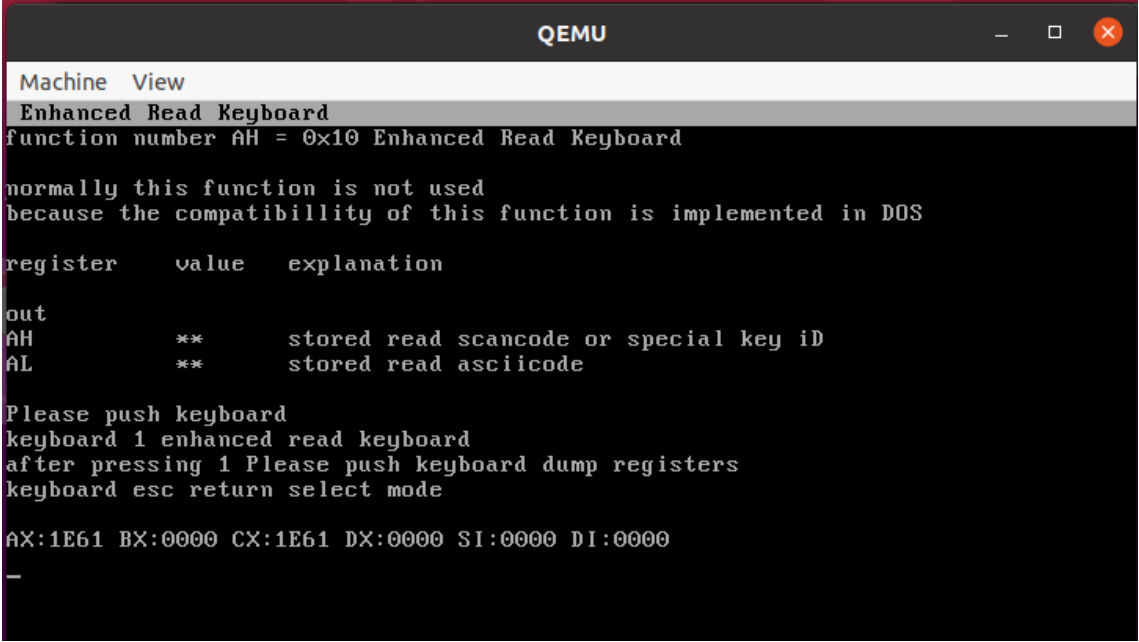
```
Machine View
Enhanced Read Keyboard
function number AH = 0x10 Enhanced Read Keyboard

normally this function is not used
because the compatibillity of this function is implemented in DOS

register    value    explanation
out
AH          **      stored read scancode or special key iD
AL          **      stored read asciiicode

Please push keyboard
keyboard 1 enhanced read keyboard
after pressing 1 Please push keyboard dump registers
keyboard esc return select mode
—
```

図 5.6.12 int16 キーボード 7 の初期画面



```
Machine View
Enhanced Read Keyboard
function number AH = 0x10 Enhanced Read Keyboard

normally this function is not used
because the compatibillity of this function is implemented in DOS

register    value    explanation
out
AH          **      stored read scancode or special key iD
AL          **      stored read asciiicode

Please push keyboard
keyboard 1 enhanced read keyboard
after pressing 1 Please push keyboard dump registers
keyboard esc return select mode

AX:1E61 BX:0000 CX:1E61 DX:0000 SI:0000 DI:0000
-
```

図 5.6.13 int16 キーボード 7 追加機能の実行結果(キーボード 1 a 入力)

5.6.8 Enhanced Read Keyboard Flags

Enhanced Read Keyboard Flags は拡張キーボードフラグ読み込みファンクションである。101 キーのシフトキーなどのステータスを読み込む。表 5.6.3 にキーの概要について示す。動作は 5.6.3 と同じである。アプリケーションによる実行は、選択画面から、キーボード 8 を入力することによって実行可能になる。追加機能は選択画面の遷移だけである。図 5.6.14 にキーボード 8 の実行結果を示す。

表 5.6.3 キーのステータス概要

AL レジスタに出力される値	概要
10000000:00000000b	SYSREQ キーが押されている状態
01000000:00000000b	キャプスロックキーが押されている状態
00100000:00000000b	ナムロックキーが押されている状態
00010000:00000000b	スクロールロックキーが押されている状態
00001000:00000000b	右 ALT キーが押されている状態
00000100:00000000b	右 CTRL キーが押されている状態
00000010:00000000b	左 ALT キーが押されている状態
00000001:00000000b	左 CTRL キーが押されている状態
00000000:10000000b	インサートモードがアクティブ
00000000:01000000b	キャプスロックキーがアクティブ
00000000:00100000b	ナムロックキーがアクティブ
00000000:00010000b	スクロールロックがアクティブ
00000000:00001000b	ALT キーが押されている状態
00000000:00000100b	CTRL キーが押されている状態
00000000:00000010b	左シフトキーが押されている状態
00000000:00000001b	右シフトキーが押されている状態

```
QEMU
Machine View
Enhanced Read Keyboard Flags
function number AH = 0x12 Enhanced Read Keyboard Flags

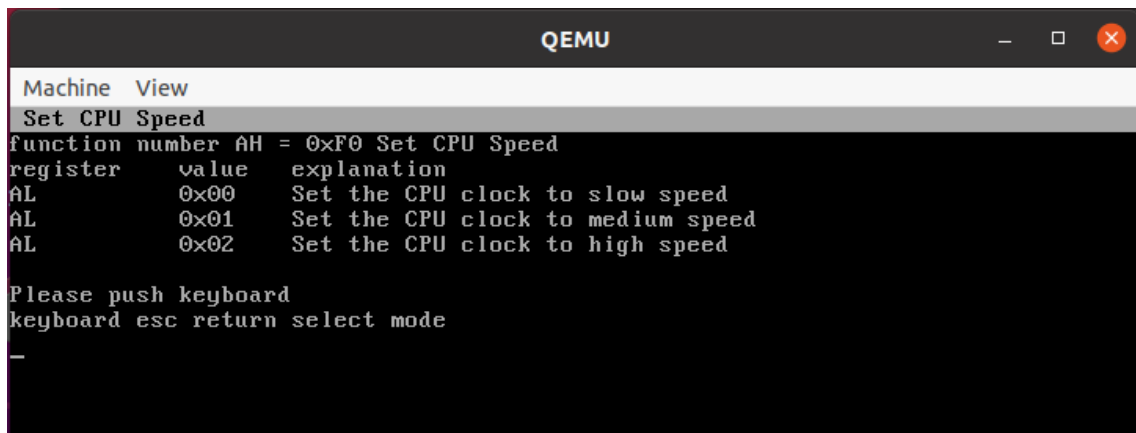
out
register    value                explanation
AX          10000000:00000000b  pushed SYSREQ key
AX          01000000:00000000b  pushed capslock key
AX          00100000:00000000b  pushed numlock key
AX          00010000:00000000b  pushed scroll lock key
AX          00001000:00000000b  pushed right alt key
AX          00000100:00000000b  pushed right ctrl key
AX          00000010:00000000b  pushed left alt key
AX          00000001:00000000b  pushed left ctrl key
AX          00000000:10000000b  pushed insert key
AX          00000000:01000000b  active capslock key
AX          00000000:00100000b  active numlock key
AX          00000000:00010000b  active scroll lock key
AX          00000000:00001000b  pushed alt key
AX          00000000:00000100b  pushed ctrl key
AX          00000000:00000010b  pushed left shift key
AX          00000000:00000001b  pushed right shift key

Please push keyboard
keyboard esc return select mode
```

図 5.6.14 キーボード 8 の実行結果

5.6.9 Set CPU Speed

Set CPU Speed は CPU クロック設定である。CPU のクロックの上げ下げが可能であり、低速、中速、高速に設定できる。追加機能は選択画面の遷移だけである。アプリケーションによる実行は、選択画面から、キーボード 9 を入力することによって実行可能になる。追加機能は選択画面の遷移だけである。図 5.6.15 にキーボード 9 の実行結果を示す。

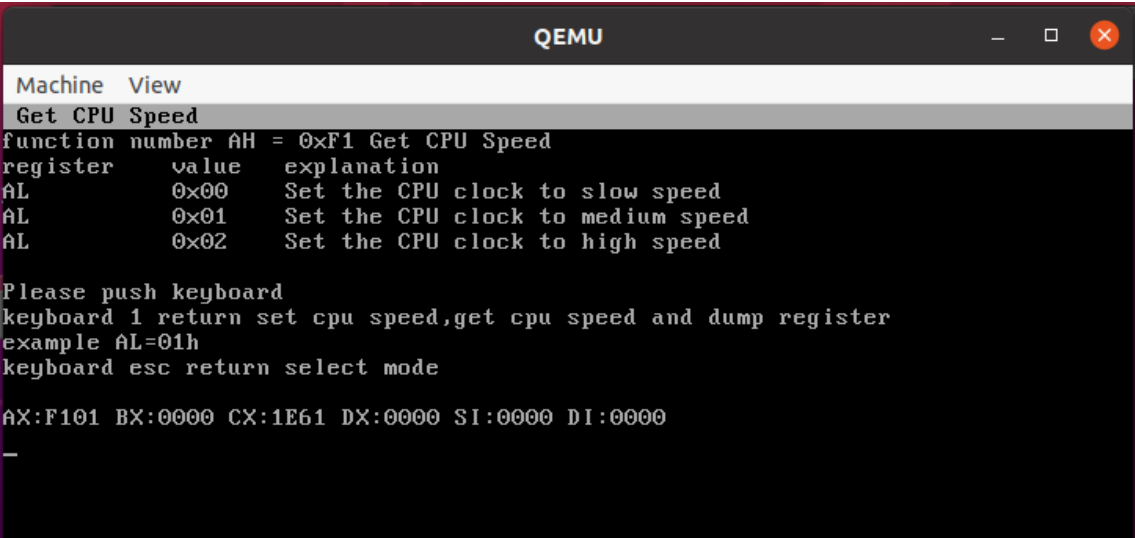


```
QEMU
Machine View
Set CPU Speed
function number AH = 0xF0 Set CPU Speed
register    value    explanation
AL         0x00     Set the CPU clock to slow speed
AL         0x01     Set the CPU clock to medium speed
AL         0x02     Set the CPU clock to high speed
Please push keyboard
keyboard esc return select mode
—
```

図 5.6.15 キーボード 9 の実行結果

5.6.10 Get CPU Speed

Get CPU Speed は CPU クロック設定取得である。CPU クロックで設定した値を取得する。アプリケーションによる実行は、選択画面から、キーボード a を入力することによって実行可能になる。図 5.6.16 にキーボード a の初期画面を示す。追加機能はレジスタの表示である。例は低速の CPU クロック設定となっている。図 5.6.17 に追加機能の実行結果を示す。

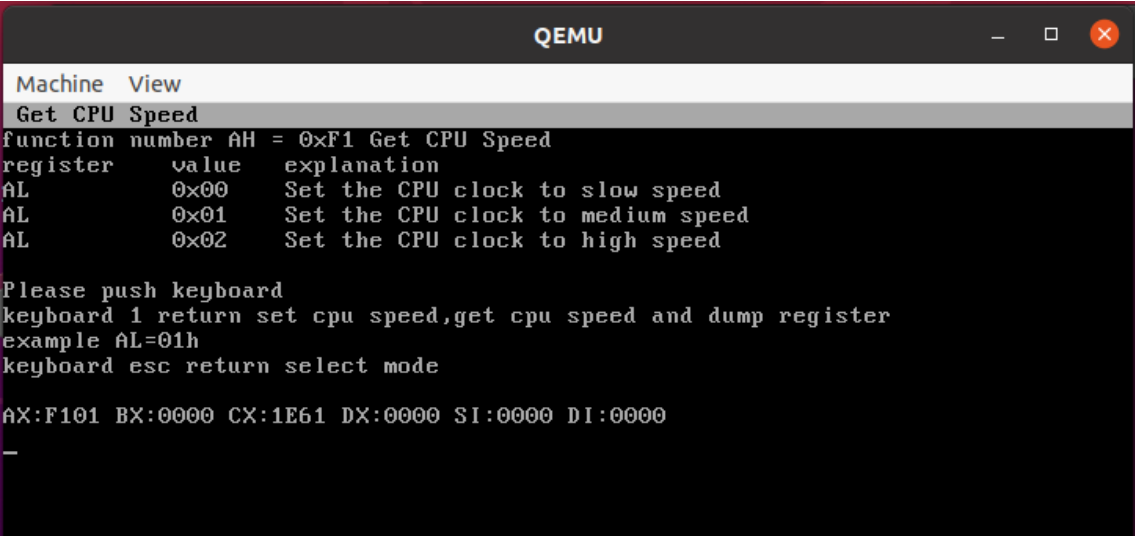


```
QEMU
Machine View
Get CPU Speed
function number AH = 0xF1 Get CPU Speed
register    value  explanation
AL         0x00   Set the CPU clock to slow speed
AL         0x01   Set the CPU clock to medium speed
AL         0x02   Set the CPU clock to high speed

Please push keyboard
keyboard 1 return set cpu speed,get cpu speed and dump register
example AL=01h
keyboard esc return select mode

AX:F101 BX:0000 CX:1E61 DX:0000 SI:0000 DI:0000
-
```

図 5.6.16 int16 キーボード a の初期画面



```
QEMU
Machine View
Get CPU Speed
function number AH = 0xF1 Get CPU Speed
register    value  explanation
AL         0x00   Set the CPU clock to slow speed
AL         0x01   Set the CPU clock to medium speed
AL         0x02   Set the CPU clock to high speed

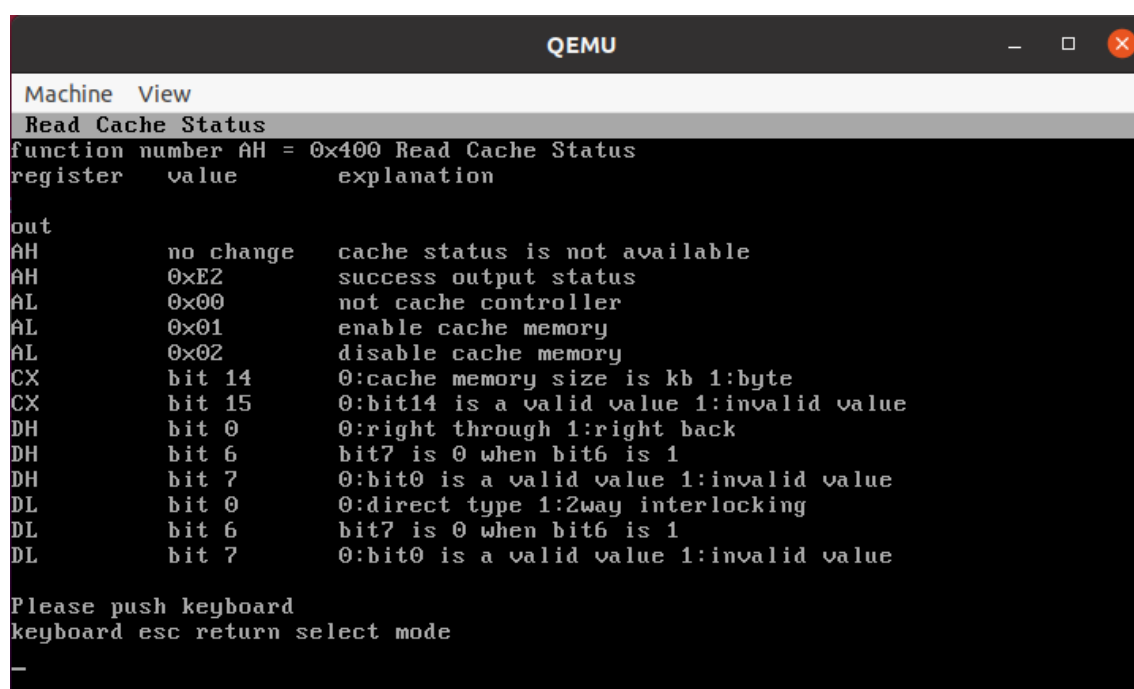
Please push keyboard
keyboard 1 return set cpu speed,get cpu speed and dump register
example AL=01h
keyboard esc return select mode

AX:F101 BX:0000 CX:1E61 DX:0000 SI:0000 DI:0000
-
```

図 5.6.17 int16 キーボード a 追加機能の実行結果(キーボード 1)

5.6.11 Read Cache Status

Read Cache Status は、キャッシュステータス読み込みである。BIOS がサポートしている外部キャッシュのステータスを読み込む。アプリケーションによる実行は、選択画面から、キーボード b を入力することによって実行可能になる。追加機能は、選択画面の遷移だけである。図 5.6.18 にキーボード b の実行結果を示す。

The image is a screenshot of a QEMU window titled "QEMU". Inside the window, the BIOS "Machine View" is displayed. The screen shows the "Read Cache Status" function, which has been executed (AH = 0x400). Below the title, there is a table with three columns: "register", "value", and "explanation". The table lists various registers and their values, along with explanations for each. At the bottom of the screen, there is a prompt "Please push keyboard" and "keyboard esc return select mode".

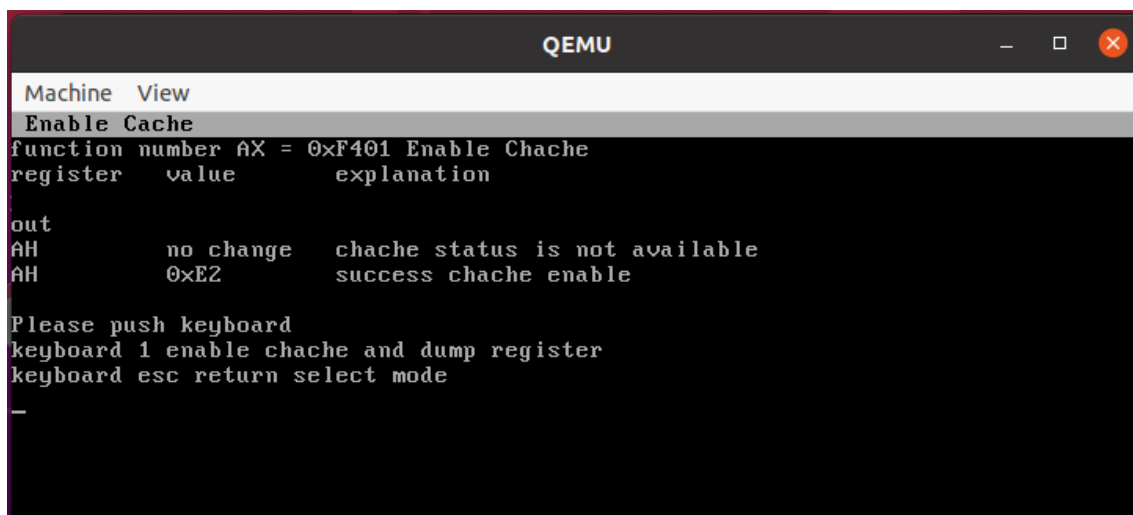
register	value	explanation
out		
AH	no change	cache status is not available
AH	0xE2	success output status
AL	0x00	not cache controller
AL	0x01	enable cache memory
AL	0x02	disable cache memory
CX	bit 14	0:cache memory size is kb 1:byte
CX	bit 15	0:bit14 is a valid value 1:invalid value
DH	bit 0	0:right through 1:right back
DH	bit 6	bit7 is 0 when bit6 is 1
DH	bit 7	0:bit0 is a valid value 1:invalid value
DL	bit 0	0:direct type 1:2way interlocking
DL	bit 6	bit7 is 0 when bit6 is 1
DL	bit 7	0:bit0 is a valid value 1:invalid value

Please push keyboard
keyboard esc return select mode

図 5.6.18 キーボード b 実行結果

5.6.12 Enable Cache

Enable Cache は、キャッシュ有効である。BIOS がサポートしている外部キャッシュを有効化する。AH レジスタで有効化されたかどうかを判断可能で、AH レジスタに 0xE2 が出力されれば、キャッシュが有効になったということになる。利用不可であれば、入力から変更はない。アプリケーションによる実行は、選択画面から、キーボード c を入力することによって実行可能になる。図 5.6.19 にキーボード c の初期画面を示す。追加機能は、キャッシュの有効化とレジスタの表示である。図 5.6.20 に追加機能の実行結果を示す。この場合、AX レジスタに F401 と出力されているので、変更は無いということなので、キャッシュの有効化に失敗したことになる。



```
QEMU
Machine View
Enable Cache
function number AX = 0xF401 Enable Cache
register    value      explanation
out
AH          no change  cache status is not available
AH          0xE2       success cache enable

Please push keyboard
keyboard 1 enable chache and dump register
keyboard esc return select mode
—
```

図 5.6.19 int16 キーボード c の初期画面

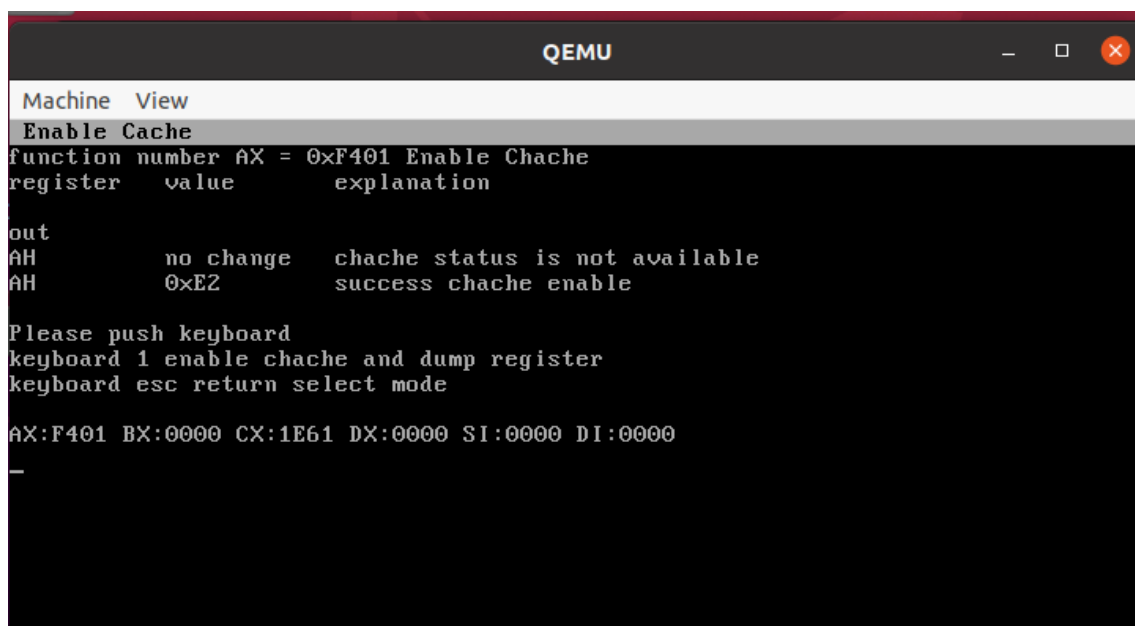
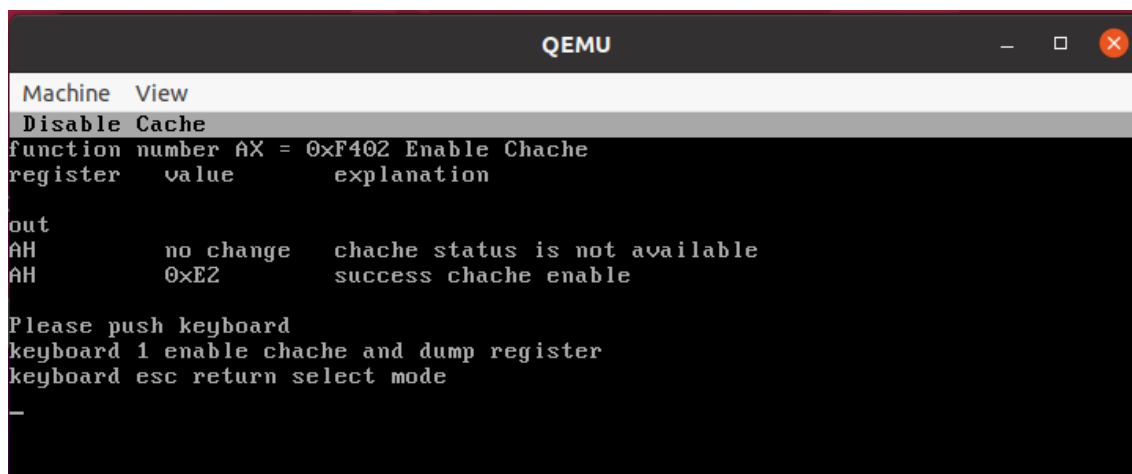


図 5.6.20 int16 キーボード c 追加機能の実行結果(キーボード 1)

5.6.13 Disable Cache

Disable Cache は、キャッシュ無効である。BIOS がサポートしている外部キャッシュを無効にする。AH レジスタで無効化されたかどうかを判断可能で、AH レジスタに 0xE2 が出力されれば、キャッシュが無効になったということになる。利用不可であれば、入力から変更はない。アプリケーションによる実行は、選択画面から、キーボード d を入力することによって実行可能になる。図 5.6.21 にキーボード d の初期画面を示す。追加機能は、キャッシュの無効化とレジスタの表示である。図 5.6.22 に追加機能の実行結果を示す。この場合、AX レジスタに F402 と出力されているので、変更は無いということなので、キャッシュの有効化に失敗したことになる。



```
QEMU
Machine View
Disable Cache
function number AX = 0xF402 Enable Cache
register    value    explanation
out
AH          no change  cache status is not available
AH          0xE2      success cache enable
Please push keyboard
keyboard 1 enable cache and dump register
keyboard esc return select mode
-
```

図 5.6.21 int16 キーボード d の初期画面

```
QEMU
Machine View
Disable Cache
function number AX = 0xF402 Enable Chache
register    value    explanation
out
AH          no change  chache status is not available
AH          0xE2      success chache enable

Please push keyboard
keyboard 1 enable chache and dump register
keyboard esc return select mode

AX:F402 BX:0000 CX:1E61 DX:0000 SI:0000 DI:0000
-
```

図 5.6.20 int16 キーボード d 追加機能の実行結果(キーボード 1)

5.7 bios コール 0x1A

bios コール 0x1A ビデオサービスの概要を表 5.7.1 に示す。表 5.7.1 は 0x1A によって実行可能なサービスの種類である。

表 5.8 0x1A 時刻、日付サービスの概要

機能番号	ファンクション名	概要
0x00	Read System Timer Count	システムタイマカウント値の読み込み
0x01	Write System Timer Count	システムタイマカウント値の書き込み
0x02	Read Real Time Clock Time	RTC による時刻の読み込み
0x03	Write Real Time Clock Time	RTC による時刻の書き込み
0x04	Read Real Time Clock Date	RTC による日付の読み込み
0x05	Write Real Time Clock Date	RTC による日付の書き込み
0x06	Set Real Time Clock Alarm	RTC のアラームの設定
0x07	Reset Real Time Clock Alarm	RTC のアラームのリセット

5.8 アプリケーション int1a.asm について

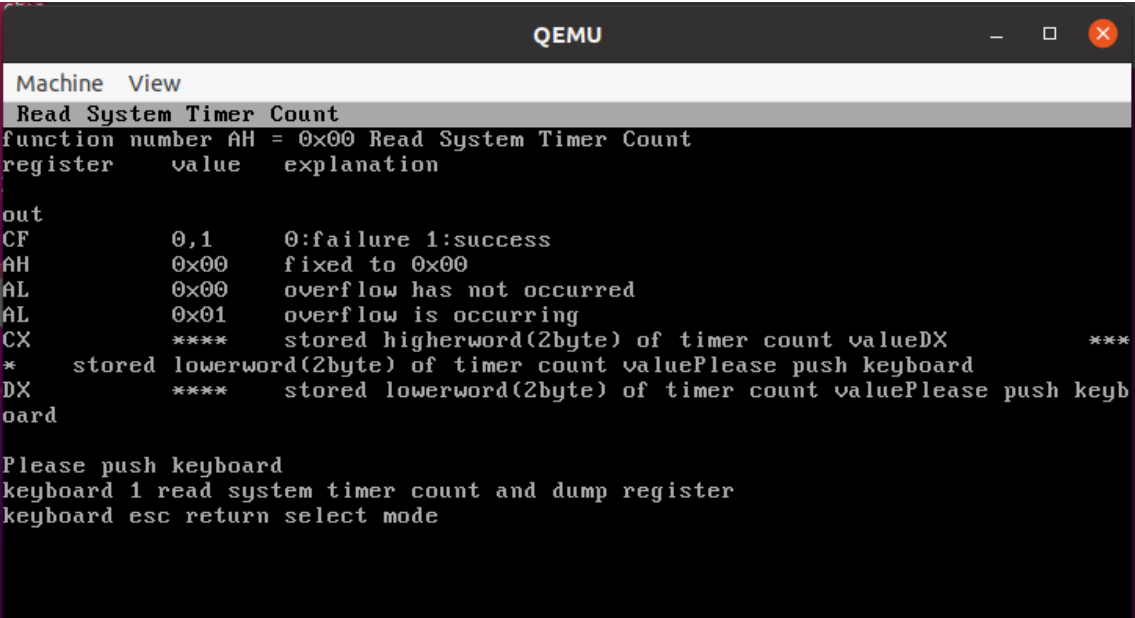
アプリケーションは、表 5.7.1 のビデオサービスを実行可能である。表 5.8.1 に割り当てたキーボードの詳細を示す。

表 5.9 0x1a 時刻、日付サービスの概要

ファンクション名	割り当てたキーボード
Read System Timer Count	1
Write System Timer Count	2
Read Real Time Clock Time	3
Write Real Time Clock Time	4
Read Real Time Clock Date	5
Write Real Time Clock Date	6
Set Real Time Clock Alarm	7
Reset Real Time Clock Alarm	8

5.8.1 Read System Timer Count

Read System Timer Count は、システムタイマカウント読み込みである。32 ビットシステムタイマカウント値を読み込む。システムタイマカウント値は CX レジスタにタイマカウント値の上位ワードが、DX レジスタにタイマカウント値の下位ワードが格納される。アプリケーションによる実行は、選択画面から、キーボード 1 を入力することによって実行可能になる。図 5.8.1 にキーボード 1 の初期画面を示す。追加機能は、命令の実行とレジスタの表示を行う。図 5.8.2 に追加機能の実行結果を示す。



```
QEMU
Machine View
Read System Timer Count
function number AH = 0x00 Read System Timer Count
register    value    explanation
out
CF          0,1      0:failure 1:success
AH          0x00      fixed to 0x00
AL          0x00      overflow has not occurred
AL          0x01      overflow is occurring
CX          ****      stored higherword(2byte) of timer count valueDX      ***
* stored lowerword(2byte) of timer count valuePlease push keyboard
DX          ****      stored lowerword(2byte) of timer count valuePlease push keyb
oard
Please push keyboard
keyboard 1 read system timer count and dump register
keyboard esc return select mode
```

図 5.8.1 int1a キーボード 1 の初期画面

```
QEMU
Machine View
Read System Timer Count
function number AH = 0x00 Read System Timer Count
register    value    explanation
out
CF          0,1      0:failure 1:success
AH          0x00      fixed to 0x00
AL          0x00      overflow has not occurred
AL          0x01      overflow is occurring
CX          ****      stored higherword(2byte) of timer count valueDX      ***
* stored lowerword(2byte) of timer count valuePlease push keyboard
DX          ****      stored lowerword(2byte) of timer count valuePlease push keyb
oard

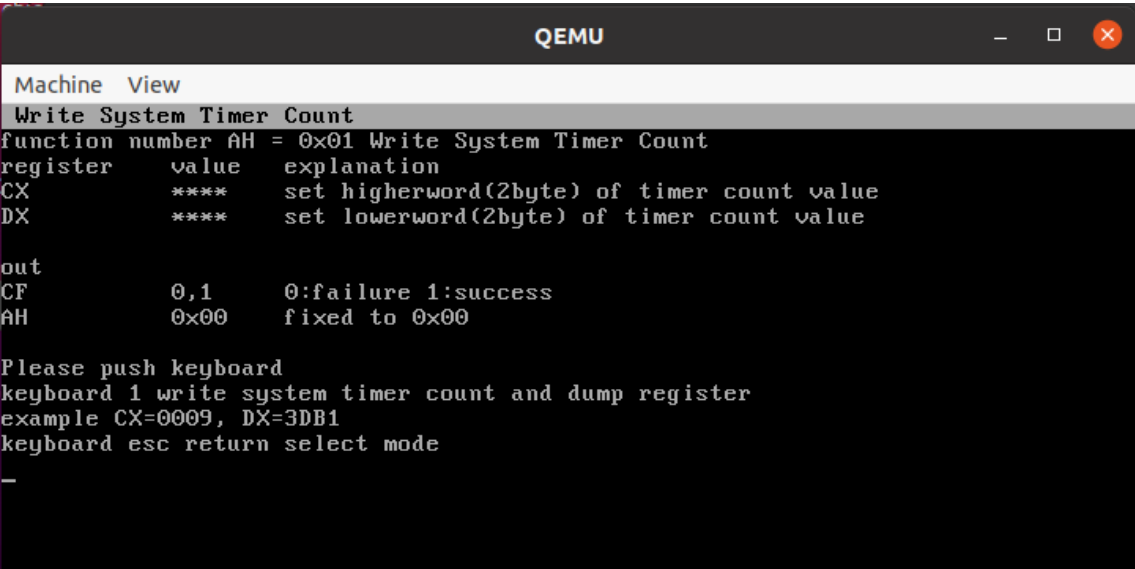
Please push keyboard
keyboard 1 read system timer count and dump register
keyboard esc return select mode

AX:0000 BX:0000 CX:0000 DX:BDC0 SI:0000 DI:0000
-
```

図 5.8.2 int1a キーボード 1 追加機能の実行結果(キーボード 1)

5.8.2 Write System Timer Count

Write System Timer Count は、システムタイマカウント書き込みである。32 ビットのシステムタイマカウントにカウント値を書き込む。CX レジスタにタイマカウント値の上位ワードを、DX レジスタにシステムタイマカウント値の下位ワードを指定できる。アプリケーションによる実行は、選択画面から、キーボード 2 を入力することによって実行可能になる。図 5.8.3 にキーボード 2 の初期画面を示す。追加機能は、命令の実行とレジスタの表示である。CX レジスタに 0009 を、DX レジスタに 3DB1 を格納している。図 5.8.4 に追加機能の実行結果を示す。



```
Machine View
Write System Timer Count
function number AH = 0x01 Write System Timer Count
register    value    explanation
CX         ****    set higherword(2byte) of timer count value
DX         ****    set lowerword(2byte) of timer count value

out
CF          0,1      0:failure 1:success
AH          0x00     fixed to 0x00

Please push keyboard
keyboard 1 write system timer count and dump register
example CX=0009, DX=3DB1
keyboard esc return select mode
-
```

図 5.8.3 int1a キーボード 2 の初期画面

```
QEMU
Machine View
Write System Timer Count
function number AH = 0x01 Write System Timer Count
register    value    explanation
CX         ****     set higherword(2byte) of timer count value
DX         ****     set lowerword(2byte) of timer count value

out
CF          0,1      0:failure 1:success
AH          0x00     fixed to 0x00

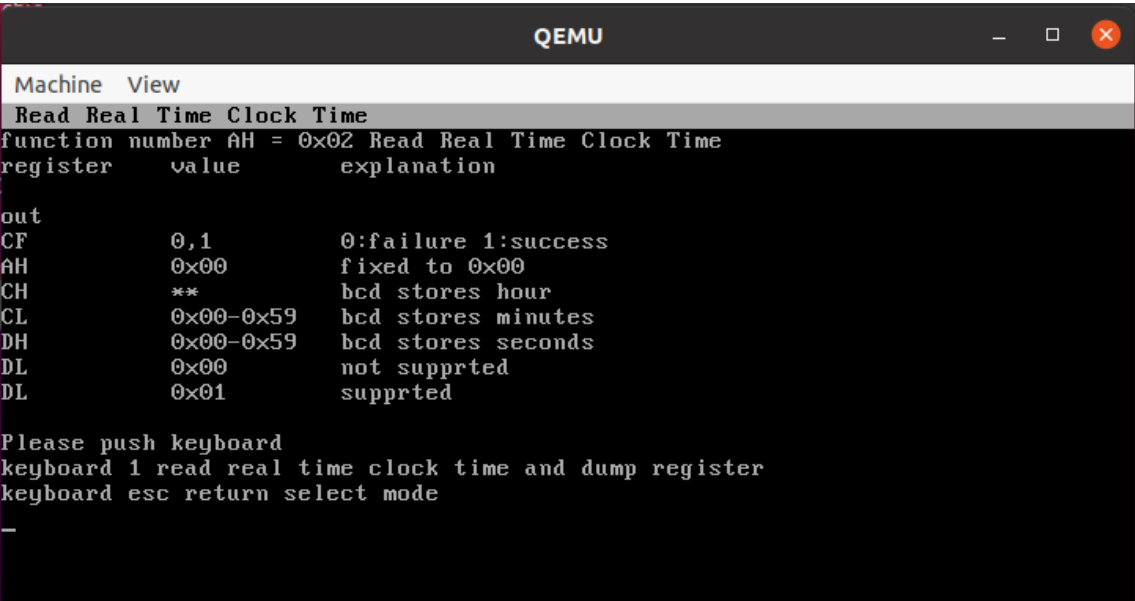
Please push keyboard
keyboard 1 write system timer count and dump register
example CX=0009, DX=3DB1
keyboard esc return select mode

AX:0031 BX:0000 CX:0009 DX:3DB1 SI:0000 DI:0000
-
```

図 5.8.4 int1a キーボード 2 追加機能の実行結果(キーボード 1)

5.8.3 Read Real Time Clock Time

Read Real Time Clock Time は、RTC 時刻読み込みである。RTC(リアルタイムクロック)から時刻情報を読み出す。命令実行時、CH レジスタに時間を、CL レジスタに分を、DH レジスタに秒が格納される。アプリケーションによる実行は、選択画面から、キーボード 3 を入力することによって実行可能になる。図 5.8.5 にキーボード 3 の初期画面を示す。追加機能は、命令の実行とレジスタの表示を行う。図 5.8.6 に追加機能の実行結果を示す。



```
QEMU
Machine View
Read Real Time Clock Time
function number AH = 0x02 Read Real Time Clock Time
register    value    explanation
out
CF          0,1      0:failure 1:success
AH          0x00      fixed to 0x00
CH          **       bcd stores hour
CL          0x00-0x59 bcd stores minutes
DH          0x00-0x59 bcd stores seconds
DL          0x00      not supprted
DL          0x01      supprted

Please push keyboard
keyboard 1 read real time clock time and dump register
keyboard esc return select mode
-
```

図 5.8.5 キーボード 3 初期画面

```
QEMU
Machine View
Read Real Time Clock Time
function number AH = 0x02 Read Real Time Clock Time
register    value    explanation
out
CF          0,1      0:failure 1:success
AH          0x00      fixed to 0x00
CH          **       bcd stores hour
CL          0x00-0x59 bcd stores minutes
DH          0x00-0x59 bcd stores seconds
DL          0x00      not supprted
DL          0x01      supprted

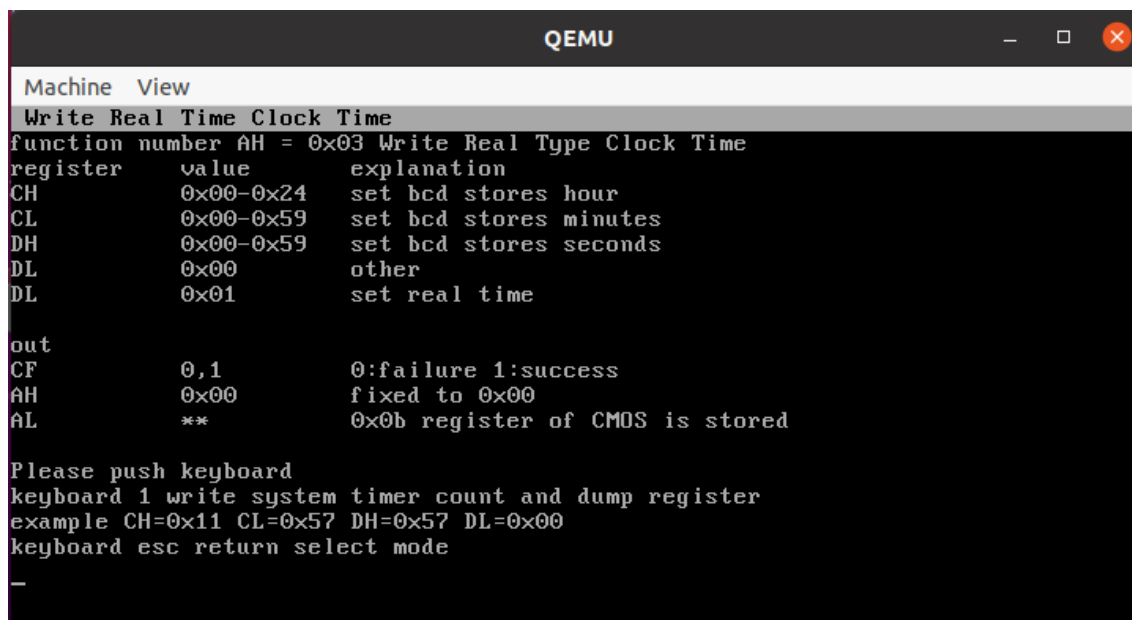
Please push keyboard
keyboard 1 read real time clock time and dump register
keyboard esc return select mode

AX:0001 BX:0000 CX:0136 DX:3700 SI:0000 DI:0000
-
```

図 5.8.6 int1a キーボード 3 追加機能の実行結果(キーボード 1)

5.8.4 Write Real Time Clock Time

Write Real Time Clock Time は、RTC 時刻書き込みである。RTC に時刻情報を書き込む。時刻を書き込む際、CH レジスタは時間を、CL レジスタは分を、DH レジスタは秒を表す。アプリケーションによる実行は、選択画面から、キーボード 4 を入力することによって実行可能になる。図 5.8.7 にキーボード 4 の初期画面を示す。追加機能は、命令の実行とレジスタの表示である。CH レジスタに 0x11 を、CL レジスタに 0x57 を、DH レジスタに 0x57 を、DL レジスタに 0x00 を格納している。図 5.8.8 に追加機能の実行結果を示す。



```
Machine View
Write Real Time Clock Time
function number AH = 0x03 Write Real Type Clock Time
register    value      explanation
CH          0x00-0x24  set bcd stores hour
CL          0x00-0x59  set bcd stores minutes
DH          0x00-0x59  set bcd stores seconds
DL          0x00      other
DL          0x01      set real time

out
CF          0,1        0:failure 1:success
AH          0x00      fixed to 0x00
AL          **        0x0b register of CMOS is stored

Please push keyboard
keyboard 1 write system timer count and dump register
example CH=0x11 CL=0x57 DH=0x57 DL=0x00
keyboard esc return select mode
-
```

図 5.8.7 キーボード 4 初期画面


```
QEMU
Machine View
Write Real Time Clock Time
function number AH = 0x03 Write Real Type Clock Time
register    value    explanation
CH          0x00-0x24  set bcd stores hour
CL          0x00-0x59  set bcd stores minutes
DH          0x00-0x59  set bcd stores seconds
DL          0x00      other
DL          0x01      set real time

out
CF          0,1        0:failure 1:success
AH          0x00      fixed to 0x00
AL          **        0x0b register of CMOS is stored

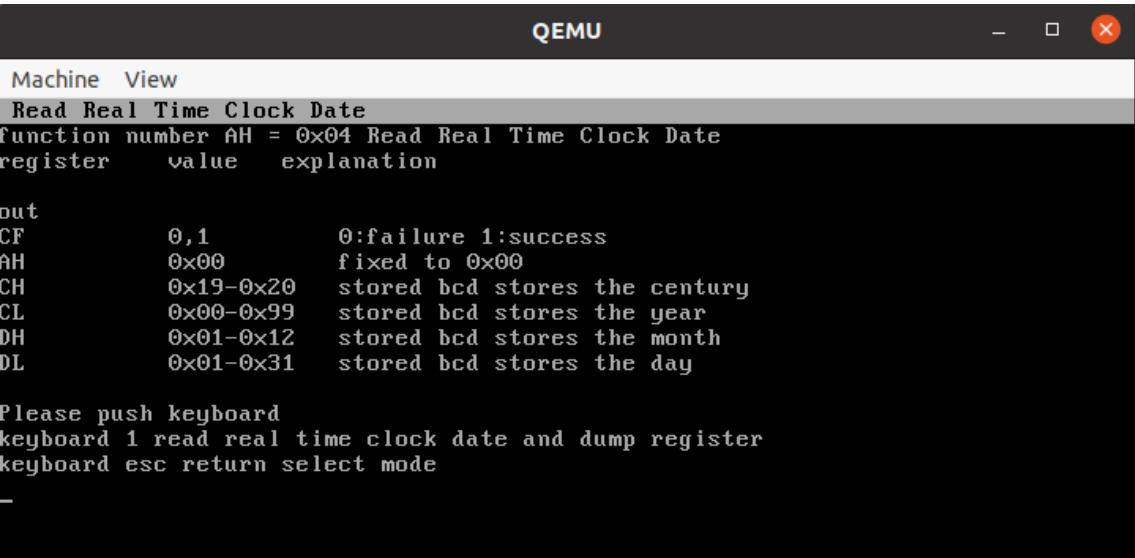
Please push keyboard
keyboard 1 write system timer count and dump register
example CH=0x11 CL=0x57 DH=0x57 DL=0x00
keyboard esc return select mode

AX:0002 BX:0000 CX:1157 DX:5700 SI:0000 DI:0000
-
```

図 5.8.8 int1a キーボード 4 追加機能の実行結果(キーボード 1)

5.8.5 Read Real Time Clock Date

Read Real Time Clock Date は、RTC 日付読み込みである。RTC により日付情報を読み出す。命令実行後、CH レジスタに世紀が、CL レジスタに年が、DH レジスタに月が、DL レジスタに日にちが BCD で格納される。アプリケーションによる実行は、選択画面から、キーボード 5 を入力することによって実行可能になる。図 5.8.9 にキーボード 5 の初期画面を示す。追加機能は、命令の実行とレジスタの表示である。図 5.8.10 に追加機能の実行結果を示す。



```
QEMU
Machine View
Read Real Time Clock Date
Function number AH = 0x04 Read Real Time Clock Date
register    value    explanation
out
CF          0,1      0:failure 1:success
AH          0x00      fixed to 0x00
CH          0x19-0x20 stored bcd stores the century
CL          0x00-0x99 stored bcd stores the year
DH          0x01-0x12 stored bcd stores the month
DL          0x01-0x31 stored bcd stores the day
Please push keyboard
keyboard 1 read real time clock date and dump register
keyboard esc return select mode
_
```

図 5.8.9 キーボード 5 初期画面

```
QEMU
Machine View
Read Real Time Clock Date
function number AH = 0x04 Read Real Time Clock Date
register    value    explanation
out
CF          0,1      0:failure 1:success
AH          0x00      fixed to 0x00
CH          0x19-0x20 stored bcd stores the century
CL          0x00-0x99 stored bcd stores the year
DH          0x01-0x12 stored bcd stores the month
DL          0x01-0x31 stored bcd stores the day

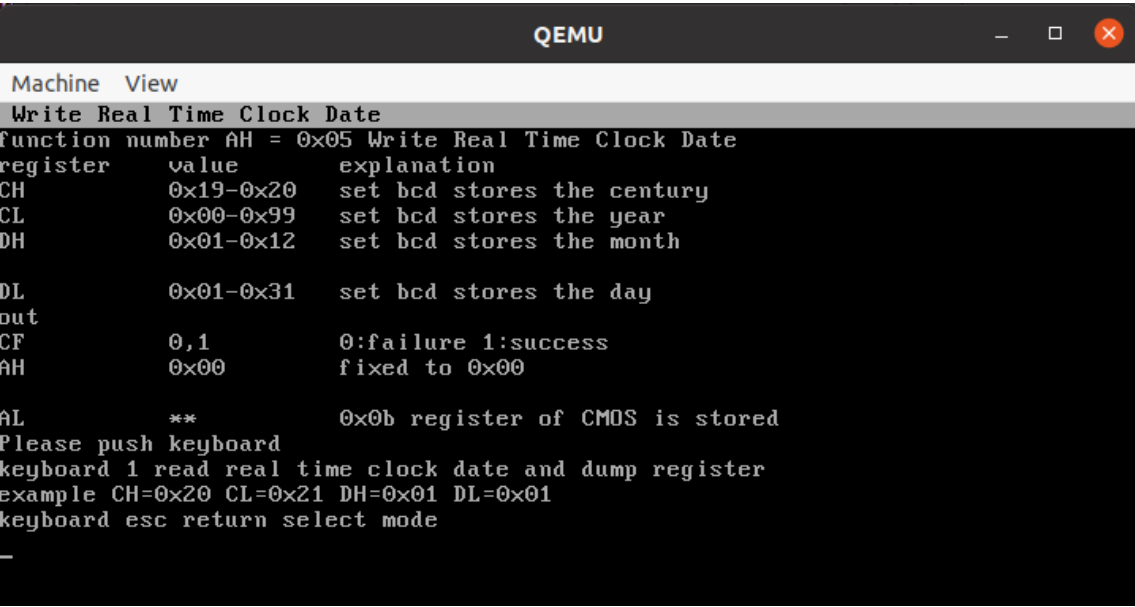
Please push keyboard
keyboard 1 read real time clock date and dump register
keyboard esc return select mode

AX:0020 BX:0000 CX:2022 DX:0226 SI:0000 DI:0000
-
```

図 5.8.10 int1a キーボード 5 追加機能の実行結果(キーボード 1)

5.8.6 Write Real Time Clock Date

Write Real Time Clock Date は RTC 日付書き込みである。RTC に日付情報を書き込む際、CH レジスタは世紀を、CL レジスタは年を、DH レジスタは月を、DL レジスタは日にちを表す。アプリケーションによる実行は、選択画面から、キーボード 6 を入力することによって実行可能になる。図 5.8.11 にキーボード 6 の初期画面を示す。追加機能は命令の実行とレジスタの表示である。CH レジスタに 0x20、CL レジスタに 0x21、DH レジスタに 0x01、DL レジスタに 0x01 を格納している。図 5.8.12 に追加機能の実行結果を示す。



```
QEMU
Machine View
Write Real Time Clock Date
function number AH = 0x05 Write Real Time Clock Date
register    value    explanation
CH          0x19-0x20 set bcd stores the century
CL          0x00-0x99 set bcd stores the year
DH          0x01-0x12 set bcd stores the month
DL          0x01-0x31 set bcd stores the day
out
CF          0,1      0:failure 1:success
AH          0x00     fixed to 0x00
AL          **       0x0b register of CMOS is stored
Please push keyboard
keyboard 1 read real time clock date and dump register
example CH=0x20 CL=0x21 DH=0x01 DL=0x01
keyboard esc return select mode
-
```

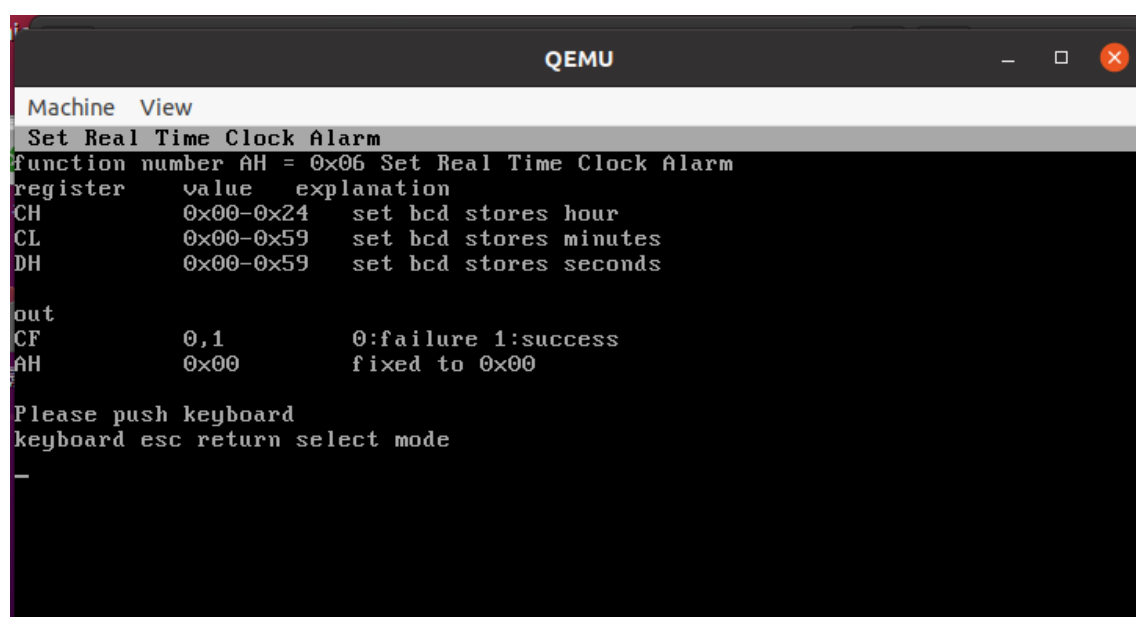
図 5.8.11 キーボード 6 初期画面

```
QEMU
Machine View
Write Real Time Clock Date
function number AH = 0x05 Write Real Time Clock Date
register    value    explanation
CH          0x19-0x20  set bcd stores the century
CL          0x00-0x99  set bcd stores the year
DH          0x01-0x12  set bcd stores the month
DL          0x01-0x31  set bcd stores the day
out
CF          0,1        0:failure 1:success
AH          0x00        fixed to 0x00
AL          **          0x0b register of CMOS is stored
Please push keyboard
keyboard 1 read real time clock date and dump register
example CH=0x20 CL=0x21 DH=0x01 DL=0x01
keyboard esc return select mode
AX:0002 BX:0000 CX:2021 DX:0101 SI:0000 DI:0000
-
```

図 5.8.12 int1a キーボード 6 追加機能の実行結果(キーボード 1)

5.8.7 Set Real Time Clock Date

Set Real Time Clock Date は、RTC アラーム設定ファンクションである。RTC にアラームの設定を行う。CH レジスタに時間、CL レジスタに分、DH レジスタに秒としてアラームを設定できる。アプリケーションによる実行は、選択画面から、キーボード 7 を入力することによって実行可能になる。追加機能は選択画面の遷移だけである。図 5.8.12 にキーボード 7 の実行結果を示す。



```
QEMU
Machine View
Set Real Time Clock Alarm
function number AH = 0x06 Set Real Time Clock Alarm
register    value    explanation
CH          0x00-0x24 set bcd stores hour
CL          0x00-0x59 set bcd stores minutes
DH          0x00-0x59 set bcd stores seconds

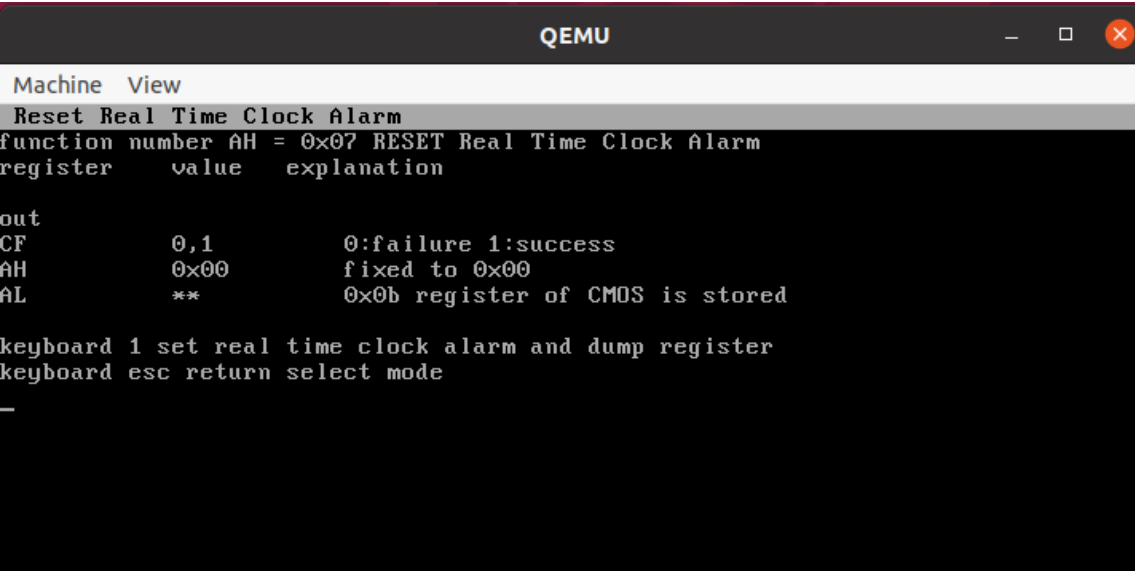
out
CF          0,1      0:failure 1:success
AH          0x00      fixed to 0x00

Please push keyboard
keyboard esc return select mode
-
```

図 5.8.13 キーボード 7 初期画面

5.8.8 Reset Real Time Clock Alarm

Reset Real Time Clock Alarm は、RTC アラームリセットである。RTC に設定されているアラームをリセットする。アプリケーションによる実行は、選択画面から、キーボード 8 を入力することによって実行可能になる。図 5.8.14 にキーボード 8 の初期画面を示す。追加機能は、命令の実行とレジスタの表示である。命令実行時、正常に動作しているなら、EFLAGS の CF ビットが 0、AH レジスタに 0x00、AL レジスタに CMOS の 0x0B レジスタに格納された値が格納される。図 5.8.15 に追加機能の実行結果を示す。



```
QEMU
Machine View
Reset Real Time Clock Alarm
function number AH = 0x07 RESET Real Time Clock Alarm
register    value    explanation
out
CF          0,1      0:failure 1:success
AH          0x00     fixed to 0x00
AL          **       0x0b register of CMOS is stored
keyboard 1 set real time clock alarm and dump register
keyboard esc return select mode
```

図 5.8.14 キーボード 8 初期画面

```
QEMU
Machine View
Reset Real Time Clock Alarm
function number AH = 0x07 RESET Real Time Clock Alarm
register    value    explanation
out
CF          0,1      0:failure 1:success
AH          0x00      fixed to 0x00
AL          **        0x0b register of CMOS is stored
keyboard 1 set real time clock alarm and dump register
keyboard esc return select mode
AX:0002 BX:0000 CX:0000 DX:0000 SI:0000 DI:0000
-
```

図 5.8.15 int1a キーボード 8 追加機能の実行結果(キーボード 1)

第六章

まとめと展望

本章の構成

- 6.1 まとめ
- 6.2 今後の展望について

6.1 オペレーティングシステムの開発について

本研究は、学習用として優れたオペレーティングシステムの作成を目的として開発を行った結果、BIOS コールは、豊富なサービスが提供されていること。BIOS コールにより比較的簡単にハードウェア制御を実行可能であること。オペレーティングシステムによるプログラムの管理には利便性があるということが分かった。

6.2 今後の展望について

今後の展望として、更に bios コールを追加すること、ソースコードの簡略化、実際に学習用としての効果を測定することである。b

付録 1 mikeos のソースコード解析 bootload.asm

付録は mikeos ソースコードの読解のヒントとして扱って欲しい。表 ex-1.1、ex-1.2 に bootload.asm の変数表、表 ex-1.3 に bootload.asm の関数表を示す。

変数名	内容	詳細
OEMLabel	db "MIKEBOOT"	ディスクラベル
BytesPerSector	dw 512	セクターあたりのバイト数
SectorsPerCluster	db 1	クラスタあたりのセクター
ReservedForBoot	dw 1	ブートレコード用に予約されたセクター
NumberOfFats	db 2	FAT のコピー数
RootDirEntries	dw 224	ルートディレクトリのエントリ数 ($224 * 32 = 7168 = 14$ セクターを読み取る)
LogicalSectors	dw 2880	論理セクターの数
MediumByte	db 0F0h	中程度の記述子バイト
SectorsPerFat	dw 9	FAT あたりのセクター
SectorsPerTrack	dw 18	トラックあたりのセクター (36 / シリンダー)
Sides	dw 2	サイド ヘッドの数
HiddenSectors	dd 0	隠れたセクターの数
LargeSectors	dd 0	LBA セクターの数
DriveNo	dw 0	ドライブ番号：0
Signature	db 41	ドライブの署名：フロッピーの場合は 41
VolumeID	dd 00000000h	ボリューム ID：任意の番号
VolumeLabel	db "MIKEOS "	ボリュームラベル：任意の 11 文字
FileSystem	db "FAT12 "	ファイルシステムの種類：変更しないで！
kern_filename	db "KERNEL BIN"	MikeOS カーネルファイル名
disk_error	db "Floppy error! Press any	エラー表示文字列 フロッピー

	key...", 0	ー
file_not_found	db "KERNEL.BIN not found!", 0	エラー表示文字列 kernel.bin
bootdev	db 0	起動デバイス番号
cluster	dw 0	ロードしたいファイルのクラ スター
pointer	dw 0	カーネルをロードするための バッファへのポインタ

表 ex-1.1 bootload.asm 変数表

変数名	初期定義位置（行数）	記述範囲（ソリューション全体）
OEMLabel	24	定義のみ
BytesPerSector	25	定義のみ
SectorsPerCluster	26	194 コメント
ReservedForBoot	27	78 コメント
NumberOfFats	28	78 コメント
RootDirEntries	29	79 コメント
LogicalSectors	31	定義のみ
MediumByte	32	定義のみ
SectorsPerFat	33	78 コメント
SectorsPerTrack	34	68 bootloader_start ラベル。 313,319 12hts ラベル
Sides	35	71 bootloader_start ラベル。 321 12hts ラベル ※現ドキュ
HideenSectors	36	定義のみ
LargeSectors	37	定義のみ
DriveNo	38	定義のみ
Signataure	39	8 のコメント 350 のコメント
VolumeID	40	定義のみ
VolumeLabel	41	定義のみ
FileSystem	42	定義のみ
kern_filename	336	125 の mov 命令
disk_error	338	173 の mov 命令

file_not_found	339	138 の mov 命令
bootdev	341	63,261,297,328 の jmp 命令。 293 の reser 命令。どちらも [] 表記
cluster	342	9,26,143,144,192,193,194,195,219,230,235,237,238 がコメント。145,198,225,250mov 命令。[] 表記
pointer	343	128,255 コメント 205 mov 命令 255 add 命令

表 ex-1.2 bootload.asm 変数表

関数名（ラベル）	詳細	初期定義位置 （行数）	記述範囲（ソリューション全体）
bootloader_start:	最初に実行される関数	48	15 jmp 命令 short
no_change:	eax の値を 0 に	73	62 je 比較
fatal_disk_error:	エラー関数 ディスク	171	66 jc 比較
floppy_ok:	データの最初のブロックを読み取る	82	82 定義のみ
read_root_dir:	BIOS を使用し、セクターを読み取る	97	106 jnc 比較
search_dir:	全てのエントリを検索	111	104 jnc 比較
next_root_entry:	カーネルファイルを 検索しバンプする	122	136 loop 命令
found_file_to_load:	クラスターをフェッチし、FAT を RAM にロード	143	123 je 命令
read_fat:	BIOS 13h 命令	159	168 jnc 命令
read_fat_ok:	カーネルをロードするセグメント フロッピー読み取り パラメータの設定	178	166 jnc 命令

load_file_sector:	セクターの論理返 還、バッファの設定	197	216,256 jmp 命令
calculate_next_cluster:	クラスターの削除、 ドロップ判定	224	213 jnc 比較
add:	4 ビットシフトアウト	240	240 定義のみ
even	最後の 4 ビットをマ スク	245	237 jz 命令
next_cluster_cont:	バッファポインタを 1 セクター長に増や す	249	242 jmp 命令 short
end:	ロードされたカーネ ルの エントリポイントに ジャンプ	259	253 jae 比較
reboot:	BIOS 16h,19h 命令 により再起動 19h が再起動命令	269	108,140,175 jmp 命 令
print_string:	SI の文字列を画面に 出力	276	139,174 call 命令
.repeat:(A20)	文字列から文字を取 得。 文字列の終わりの判 定	281	286 jmp 命令
.done:(A20)	終了を示すローカル 関数	288	284 je 比較
reset_floppy:	フロッピーのリセッ ト	293	105,167,215 call 命 令
l2hts:	int13 のヘッド、ト ラック、 セクターの設定を計 算する	305	84,148,201 call 命 令
buffer:	ディスクバッファの 開始 (8K スタック の開始)	353	86,116,132,150,231 mov 命令

表 ex-1.3 bootload.asm 変数表

付録2 mikeos のソースコード解析 kernel.asm

続いて、kernel.asm である。表 ex-2.1、ex-2.2 に変数表を、ex-2.3 に関数表を、ex-2.4 に call 関数表を示す。

変数名	内容	関数
os_init_msg	db 'Welcome to MikeOS', 0	option_screen:
os_version_msg	db 'Version ', MIKEOS_VER, 0	option_screen:
dialog_string_1	db 'Thanks for trying out MikeOS!', 0	option_screen:
dialog_string_2	db 'Please select an interface: OK for the', 0	option_screen:
dialog_string_3	db 'program menu, Cancel for command line.', 0	option_screen:
kern_file_name	db 'KERNEL.BIN', 0	not_bas_extension:
autorun_bin_file_name	db 'AUTORUN.BIN', 0	not_bas_extension:
autorun_bas_file_name	db 'AUTORUN.BAS', 0	not_bas_extension:
bin_ext	db 'BIN'	not_bas_extension:
bas_ext	db 'BAS'	not_bas_extension:
kerndlg_string_1	db 'Cannot load and execute MikeOS kernel!', 0	not_bas_extension:
kerndlg_string_2	db 'KERNEL.BIN is the core of MikeOS, and', 0	not_bas_extension:
kerndlg_string_3	db 'is not a normal program.', 0	not_bas_extension:
ext_string_1	db 'Invalid filename extension! You can', 0	not_bas_extension:
ext_string_2	db 'only execute .BIN or .BAS programs.', 0	not_bas_extension:
program_finished_msg	db '>>> Program finished --- press a key to continue...', 0	not_bas_extension:
fmt_12_24	db 0	not_bas_extension:

fmt_date	db 0, '/'	not_bas_extension:
----------	-----------	--------------------

表 ex-2.1 kernel.asm 変数表

変数名	詳細	初期定義位置 (行数)	記述範囲 (ソリューション全体)
os_init_msg	スクリーン設定	201	179,211 mov 命令
os_version_msg	バージョン文字列	202	180,212 mov 命令
dialog_string_1	アプリセクター、 コマンドラインを必要としているか尋ねる	204	184 mov 命令
dialog_string_2	インターフェイスの 選択	205	185 mov 命令
dialog_string_3	プログラムメニュー コマンドラインのキャンセル	206	186 mov 命令
kern_file_name	os_string_compare 用変数	345	223 mov 命令
autorun_bin_file_name	os_file_exists 用変数	347	150 mov 命令
autorun_bas_file_name	A9 BAS 版	348	定義のみ
bin_ext	判定用変数	350	244 mov 命令
bas_ext	判定用変数	351	305 mov 命令
kerndlg_string_1	エラー カーネルロード	353	280 mov 命令
kerndlg_string_2	kernel.bin の説明	354	281 mov 命令
kerndlg_string_3	エラー 通常プログラム判定	355	282 mov 命令
ext_string_1	エラー ファイル名 拡張子	357	334 mov 命令
ext_string_2	bin,bas のみの実行	358	335 mov 命令
program_finished_msg	basic プログラム終了 の報告	360	271,323 mov 命令

fmt_12_24	ゼロ以外 = 24 時間 形式	369	定義のみ
fmt_date	0、1、2 = M / D / Y、D / M / Y または Y / M / D ビット 7 = 名前を数 か月間使用 ビット 7 = 0 の場 合、2 番目のバイト = 区切り文字	371	定義のみ

表 ex-2.2 kernel.asm 変数表

関数名 (ラベル)	詳細	初期定義位 置 (行数)	記述範囲 (ソリューション全 体)	関数内で call される 関数
os_main:	メインカー ネルコード の開始	110	35 jmp 命令 ベクタ ー	無し
no_change:	os_main の if 処理	139	127 je 比較	os_seed_random os_load_file os_file_exists
no_autorun_bin:	autorun の 有り無し判 定	161	152 jc 比較	os_file_exists os_load_file os_clear_screen os_run_basic
option_screen:	スクリーン の開始	178	164,220 jc 比較 196 jmp 命令	os_draw_background os_dialog_box os_clear_screen os_command_line
app_selector:	メイン画面 のレイアウ ト ファイル呼 び出し	210	172,276,286,328,340 jmp 比較 191 jne 比較	os_draw_background os_file_selector os_string_compare os_string_length os_load_file

execute_bin_program:	画面のクリア	257	156 jmp 比較	os_clear_screen 32768 ※定数 os_print_string os_wait_for_key os_clear_screen
no_kernel_execute:	カーネル実行の警告、判定	279	225 jc 比較	os_dialog_box
not_bin_extension:	bin の判定	289	247 jne 比較	os_string_length os_load_file os_clear_screen os_run_basic os_print_string os_wait_for_key os_clear_screen
not_bas_extension:	bas の判定	331	308 jne 比較	os_dialog_box

表 ex-2.3 kernel.asm 関数表

call 関数名	詳細	定義ファイル名 位置
os_command_line	コマンドライン入力モード	sources/features/cli.asm 9
os_seed_random	乱数の生成	source/features/math.asm 12
os_load_file	ファイルを RAM にロードする p4	source/features/disk.asm 146
os_file_exists	フロッピーにファイルが存在するかどうかを確認する	sources/features/disk.asm 654
os_print_string	テキストの表示	sources/features/screen.asm 13
os_clear_screen	画面を背景にクリア	sources/features/screen.asm 35
os_file_selector	ファイル選択ダイアログを表示する	sources/features/screen.asm 183
os_draw_background	テキストを含む白い上部と下部のバー、および色付きの中央セクションを備えたクリアな画面	sources/features/screen.asm 544

os_dialog_box	画面中央にボタン付きのダイアログボックス	sources/features/screen.asm 744
os_string_length	文字列の長さを返す	sources/features/string.asm 13
os_string_compare	二つの文字列が一致するかどうかを確認	sources/features/string.asm 347
32768 ※定数	定数なので無し	定数なので無し
os_wait_for_key	キーが押されるのを待って、キーを返す	sources/features/keyboard.asm 12
os_run_basic	指定されたポイントでカーネル BASIC インタプリタを実行する	sources/features/basic.asm 25

表 ex-2.4 kernel.asm call 関数表