Java アプリケーションによる CASL2 システム（ V2.0 ）

令和元年 10 月 3 日

１．はじめに

このシステムは，“アセンブラ言語の仕様”に従って記述されたアセンブラ（CASL II）のプログラムのアセンブル及び実行をサポートする Java アプリケーションです。

なお，このシステムにより提供されるプログラム CASL2 及び RUN は，Java SE 8 にて動作を検証しております。ただし，全ての PC 環境での動作を保証するものではありません。

２．システムの入手

本システムは，独立行政法人情報処理推進機構（IPA） IT 人材育成センター 国家資格・試験部のホームページ（http://www.jitec.ipa.go.jp/）からダウンロードできます。ZIP 形式になっております。

３．ファイル構成

システムは，次のようなファイル構成になっております。

```
javaCASL2_2.0
├── readme.txt              使用条件
├── Manual.pdf              マニュアル
├── casl2.jar               アセンブラプログラム
├── run.jar                 実行プログラム
└── TestProgram
    ├── CASL2.bat           アセンブルするためのバッチファイル
    ├── RUN.bat             実行するためのバッチファイル
    ├── ExplanatoryNote.txt
    │                       テストプログラムの説明
    ├── 1）
    ├── ：                  } テストプログラム
    └── 10）
```

４．動作環境

本システムを利用するには，次の環境が必要です。

・Java

・テキストエディタ

・コマンドラインインタフェース

５．利用の手順

１）Java の入手とインストール

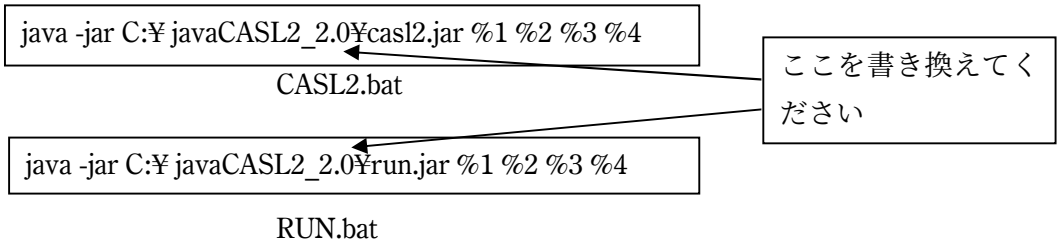
本システムの動作には Java SE 8 が必要です。Java のサイトの説明を参照し，インストールを行ってください。既に Java をインストールされている方は手順２）以降をご参照ください。

２）プログラムファイルのコピー

ダウンロードした ZIP 形式ファイルを解凍し，javaCASL2_2.0 ディレクトリを C: 直下にコピーしてください。異なる場所にコピーすることも可能ですが，その場合は，手順３）をご参照ください。コピー後は，ダウンロードした ZIP 形式ファイル及び解凍したファイルを削除してもかまいません。

３）アセンブラ・実行バッチファイル（CASL2.bat，RUN.bat）の編集

手順２）で C: 直下と異なる場所にコピーした場合は，バッチファイルの casl2.jar と run.jar のパスを変更してください。



４）動作確認

同梱のテストプログラムを用いて動作確認を行ってください。テストプログラムについては，ExplanatoryNote.txt をご参照ください。

６．アンインストール方法

コピーした javaCASL2_2.0 ディレクトリごと削除してください。Java のアンインストールについては，Java のサイトの説明をご参照ください。

７．使い方

ソースファイルのアセンブル及び実行はコマンドプロンプトなどのコマンドラインインタフェース上で行います。本書では，アセンブル及び実行には，バッチファイルを利用します。アセンブルしたいファイルと同じディレクトリに CASL2.bat と RUN.bat をコピーしてください。

１）アセンブル

指定されたソースプログラムをアセンブルし，オブジェクトプログラムを作成します。

入力形式

```
>CASL2 ソースプログラム指定 オブジェクトプログラム指定 オプション指定
```

ソースプログラム指定：

CASL2 のソースプログラムファイル名を指定します。

形式名は cas で，形式名は省略可能です。

オブジェクトプログラム指定：

CASL2 のオブジェクトプログラムファイル名を指定します。

形式名は obj です。オブジェクトプログラムファイルの指定は省略可能で，省略した場合には，ソースプログラム名の形式名を obj としてオブジェクトプログラムファイルを作成します。

オプション指定：

オプションを付けることでリストの出力を指定できます。この指定を省略した場合には，エラーのリストだけ表示されます。

-L　ラベルの情報を表示します。

-S　ソースリストを表示します。

-A　ソースリストにオブジェクトの情報を追加して表示します。

-L 及び-S を指定したことになります。

・アセンブルリストを保存したい場合

デバッグなどにおいてアセンブルリストを保存したい場合には，次のように入力します。

>CASL2　test.cas　-A　>test.lst

これで test.cas をアセンブルし，test.lst にアセンブルリストが保存されます。

使用例

>CASL2　test

　ソースプログラム test.cas をアセンブルして，オブジェクトプログラム test.obj を作成します。

>CASL2　test　-L

　ソースプログラム test.cas をアセンブルして，ラベルの情報を表示してオブジェクトプログラム test.obj を作成します。

>CASL2　test　-S

　ソースプログラム test.cas をアセンブルして，ソースリストを表示してオブジェクトプログラム test.obj を作成します。

>CASL2　test　-A

　ソースプログラム test.cas をアセンブルして，ラベルの情報及びソースリストを表示してオブジェクトプログラム test.obj を作成します。

２）実行

　指定されたオブジェクトプログラムを入力し，未定義のプログラムが存在すれば必要なオブジェクトプログラムをカレントディレクトリ内で検索して，見つかれば組み込んで実行するためのプログラムを完成させます。

　プログラムが完成後，指定に従ってプログラムを実行します。

入力形式

>RUN オブジェクトプログラム指定 オプション指定

オブジェクトプログラム指定：

　CASL2 のオブジェクトプログラムファイルを指定します。

　形式名は obj で，形式名は省略可能です。

オプション指定：

　実行のオプションについて指定します。この指定を行わない場合には，ユーザのプログラムに関する出力以外は出力されません。

　　-L　プログラムローディング終了時のプログラムの内容を出力します。

　　-M　プログラムのマップを出力します。

-T　プログラムをトレースします。

-S　プログラムを 1 ステップごとに実行します。

-Bxxxx

デバッグのための機能です。アドレス xxxx（ブレークポイント）で命令を実行する前に停止して，状況の表示や変更のための操作が可能になります。操作のコマンドには次のものがあります。

R　レジスタの内容を表示します。

M[xxxx[,yyyy]]

メモリの内容を表示します。xxxx 及び yyyy を省略すると，プログラムで使用しているすべてのメモリを表示します。

使用例

M100,200　#100 番地から#200 番地の内容を表示します。

Sxxxx[=yyyy]

メモリの内容を変更します。

使用例

S100=0200　#100 番地の内容を#0200 に変更します。

Bxxxx

ブレークポイントを変更します。

使用例

B100　#100 番地にブレークポイントを変更します。

G[xxxx]

次の実行アドレスを指定します。アドレスを省略すると停止したアドレスから実行を継続します。

使用例

G100　#100 番地から実行します。

E

プログラムの実行を中止します。

使用例

>RUN　test

オブジェクトプログラム test.obj をローディングして実行します。

>RUN　test　-T

オブジェクトプログラム test.obj をローディングして，トレースしながら実行します。

>RUN　test　-B10

オブジェクトプログラム test.obj をローディングして，10 番地にブレークポイントを設定して実行します。

>RUN　test　-S

オブジェクトプログラム test.obj をローディングして，ステップで確認しながら実行します。

８．CASL2 の追加命令

本システムには，“アセンブラ言語の仕様”に記されていないデバッグのための命令をいくつか用意してあります。

・レジスタダンプ命令　DREG

この命令が呼ばれたときに，8 文字のメッセージとレジスタの内容を表示します。スタックにデータがあればスタックの内容（先頭から 64 語）も表示します。

形式

DREG　メッセージ

メッセージ：8 文字のメッセージを指定

ラベル又はリテラルで指定します。

使用例 1

```
        DREG    MSG                 ;メッセージとレジスタの内容を表示
          ⋮
MSG     DC      '*MESSAGE'      ;メッセージ
```

使用例 2

```
        DREG    ='*MESSAGE'     ;メッセージとレジスタの内容を表示
          ⋮
```

・メモリダンプ命令　DMEM

命令が呼ばれたときに 8 文字のメッセージを表示し，指定されたメモリの内容を表示します。

形式

DMEM　メッセージ,開始アドレス,終了アドレス

メッセージ：8 文字のメッセージを指定
ラベル又はリテラルで指定します。

開始アドレス：表示するメモリの開始アドレス（ラベル）又は 10 進数
あるいは 16 進数で指定します。10 進数か 16 進数で指定した場合には絶対アドレスになります。

終了アドレス：表示するメモリの終了アドレス（ラベル）又は 10 進数
あるいは 16 進数で指定します。10 進数か 16 進数で指定した場合には絶対アドレスになります。

使用例 1

```
        DMEM   MSG,DATAS,DATAE
          ⋮
MSG     DC     '*MESSAGE'
DATAS   DC     'HELLO CASL2'
DATAE   DC     'END'
```

使用例 2

```
        DMEM   ='*MESSAGE',DATAS,DATAE
          ⋮
DATAS   DC     'HELLO CASL2'
DATAE   DC     'END'
```

使用例 1 と使用例 2 の実行例

```
* *MESSAGE *       Start:005E End:0069
 0058: ---- ---- ---- ---- ---- ---- 0048 0045              HE
 0060: 004C 004C 004F 0020 0043 0041 0053 004C    LLO CASL
 0068: 0032 0045 ---- ---- ---- ---- ---- ----     2E
```

注　表示において，各表示行の 1 カラム目は通常空白であるが，‘*’ が表示されることがある。‘*’ が表示された場合，‘*’ が表示された行と一つ前の行の間の行が省略されていることを意味し，省略行は一つ前の行のメモリ内容と同一である。

使用例 3

```
        DMEM   MSG,#100,#200
          ⋮
MSG     DC      '*MESSAGE'
```

この例では，#100 から #200 までの出力を指定しています。

・プログラム開始命令　SAVE

レジスタ退避を伴うプログラムの開始に使用する命令で，START 命令とレジスタの PUSH 命令を組み合わせたものです。プログラム終了命令 RETURN とともに用います。

形式

SAVE　ALL　or　レジスタ指定

ALL　　：GR1 から GR7 まで退避

レジスタ指定：指定されたレジスタを退避

・プログラム終了命令　RETURN

プログラム開始命令で退避されたレジスタを回復して RET します。レジスタの POP 命令と RET 命令を組み合わせたものです。

形式

RETURN

使用例

```
PROG1  SAVE            ALL      ;すべてのレジスタを退避
          ⋮
       RETURN                    ;すべてのレジスタを回復
```

```
PROG2  SAVE            GR1,GR3,GR7     ;GR1，GR3，GR7 を退避
          ⋮
       RETURN                          ; GR1，GR3，GR7 を回復
```

９．その他の制限事項

本システムでは，幾つかの制約事項を設けています。アセンブラ，実行にはそれぞれ次のような処理制限があります。

１）アセンブラプログラム casl2.jar における処理制限

・一つのソースプログラムは 1000 行以内
・一つのソースプログラム含まれるラベルは 200 個以内
・命令のオペランドの数は 40 個以内
・マクロ機能は非サポート
（IN,OUT,RPUSH 及び RPOP は命令としてサポートしています）

２）実行プログラム run.jar における処理制限

・主記憶は 60×1024 語
・組み込めるオブジェクトプログラムの数は 30 個以内
・IN 命令のファイルの終わりは ctrl-Z cr で入力
・SVC を実行すると，レジスタの内容を表示