レガシー・コード活用プロジェクト

MOS6502 エミュレータ仕様書

内容

1. 概要

本書は、レガシー・コード活用プロジェクトの一環として作成された MOS6502 エミュレータの仕様について述べる物である。

2. データ構造

本章では、MOS6502 エミュレータのデータ構造について述べる。

2.1 プログラム・レジスタ・セット

プログラム・レジスタ・セットは T_REGS 型構造体として与えられる。

T_REGS 型構造体の型定義を以下に示す。

```
typedef struct t_regs {
        unsigned char   A;
        unsigned char   Y;
        unsigned char   X;
        unsigned char   P;
} T_REGS;
```

2.2 システム・レジスタ・セット

Z80 に用意されたシステム・レジスタ・セットを構造体として提供する。

構造体の型宣言を以下に示す。

```
typedef struct t_sregs {
        t_PC_st         PC;     /* Program Counter */
        unsigned char   S;      /* Stack Pointer */
} T_SREGS;
```

プログラム・カウンタ PC は t_PC_st 型構造体の変数として宣言される。

構造体 t_PC_st 型の型定義を以下に示す。

```
typedef struct t_pc_st {
        union pc {
                t_PC_reg        t_pc;
                unsigned short  pc;
        } pc;
} t_PC_st;
```

```
typedef struct t_pc_reg {
#if defined (BIG_ENDIAN)
        unsigned char   PCH;
        unsigned char   PCL;
#else
        unsigned char   PCL;
        unsigned char   PCH;
#endif
} t_PC_reg;
```

2.3 バス

MOS6502 に実装されているコントロール・バス、データ・バス、アドレス・バスを構造体として提供する。

バス制御を提供する T_PIN 型構造体の型定義を以下に示す。

```
typedef struct t_pin {
        T_PIN_CTL       t_ctl;
        T_PIN_DATA      t_data;
        T_PIN_ADDR      t_addr;
} T_PIN;
```

ここで、T_PIN 型構造体中の各構造体は各種のバスを表す。

T_PIN 型構造体に含まれる構造体の型定義を以降に示す。

2.3.1 コントロール・バス

T_PIN_CTL 型構造体の型定義を以下に示す。

```
typedef struct t_pin_ctl {
        union ctl {
                t_CTL_bus       t_ctl;
                unsigned short  ctl;
        } ctl;
} T_PIN_CTL;
```

t_CTL_bus 型構造体の型定義を以下に示す。各メンバは MOS6502 に実装されているコントロール・バスを示している。

```
typedef struct t_ctl_bus {
        unsigned short  RDY:1;
        unsigned short  CK1:1;
        unsigned short  IRQ:1;
        unsigned short  NMI:1;
        unsigned short  SYNC:1;
        unsigned short  R_W:1;
        unsigned short  CK0:1;
        unsigned short  S0:1;
        unsigned short  CK2:1;
        unsigned short  RES:1;
        unsigned short  RFU:6;
} t_CTL_bus;
```

2.3.2 データ・バス

T_PIN_DATA 型構造体の型定義を以下に示す。

```
typedef struct t_pin_data {
        union data {
                t_DATA_bus      t_data;
                unsigned short  data_bus;
        } data;
} T_PIN_DATA;
```

t_DATA_bus 型構造体の型定義を以下に示す。各メンバは MOS6502 に実装されているデータ・バスを示している。

```
typedef struct t_data_bus {
        unsigned short  DB7:1;
        unsigned short  DB6:1;
        unsigned short  DB5:1;
        unsigned short  DB4:1;
        unsigned short  DB3:1;
        unsigned short  DB2:1;
        unsigned short  DB1:1;
        unsigned short  DB0:1;
} t_DATA_bus;
```

2.3.3 アドレス・バス

T_PIN_ADDR 型構造体の型定義を以下に示す。

```
typedef struct t_pin_addr {
        union addr {
                t_ADDR_bus      t_addr;
                unsigned short  addr_bus;
        } addr;
} T_PIN_ADDR;
```

t_ADDR_bus 型構造体の型定義を以下に示す。各メンバは Z80 に実装されているアドレス・バスを示している。

```
typedef struct t_addr_bus {
        unsigned short  AB15:1;
        unsigned short  AB14:1;
        unsigned short  AB13:1;
        unsigned short  AB12:1;
        unsigned short  AB11:1;
        unsigned short  AB10:1;
        unsigned short  AB9:1;
        unsigned short  AB8:1;
        unsigned short  AB7:1;
        unsigned short  AB6:1;
        unsigned short  AB5:1;
        unsigned short  AB4:1;
        unsigned short  AB3:1;
        unsigned short  AB2:1;
        unsigned short  AB1:1;
        unsigned short  AB0:1;
} t_ADDR_bus;
```

3. 関数

本章では、MOS6502 エミュレータで提供する関数について述べる。

3.1 ブロック転送コール・バック関数

HuC6280 のブロック転送命令について、実用性を向上させるために、コール・バック関数を実装する。

以下に、ブロック転送コール・バック関数のプロトタイプ宣言を示す。

void　callback_bt_6502(unsigned short dst, unsigned short src, unsigned char data)

引数 1	: unsigned short dst	ディスティネーション・アドレス
引数 2	: unsigned short src	ソース・アドレス
引数 3	: unsigend char data	転送データ

3.2 コンフィグ関数

コンフィグ関数は、後述のサイクル・ステート調整関数で用いる実 CPU 周波数と、エミュレート CPU 周波数を設定する。実 CPU 周波数はエミュレータを実行する PC の CPU 周波数、エミュレート CPU 周波数はエミュレートする Z80 の周波数である。

以下に、コンフィグ関数のプロトタイプ宣言を示す。

void　config_6502 (double Real_Freq, double Emu_Freq)

引数 1	: double Real_Freq	実 CPU 周波数
引数 2	: double Emu_Freq	エミュレート CPU 周波数

3.3 サイクル・ステート調整関数

サイクル・ステート調整関数は、命令のステートを調整するために用いる。

以下に、サイクル・ステート調整関数のプロトタイプ宣言を示す。

void　clk_adj_6502 (unsigned int cycle)

引数 1	: unsigned int cycle	命令のステート数

3.4 CPU リセット関数

CPU リセット関数は、バス信号/プログラム・カウンタ/フラグを初期化する。
以下に、CPU リセット関数のプロトタイプ宣言を示す。

void　　rst_6502 (T_PIN *, T_REGS *, T_SREGS *, unsigned char *)

引数 1	: T_PIN *cpu_pin	バス
引数 2	: T_REGS *reg	プログラム・レジスタ
引数 3	: T_SREGS *sysreg	システム・レジスタ
引数 4	: unsigned char *cpuflag	CPU フラグ

3.5 命令実行関数

命令実行関数は、MOS6502 のインストラクション・コードを読み込み、命令を実行する。
以下に、命令実行関数のプロトタイプ宣言を示す。

```
int
exe_6502 (      unsigned char *mem,
                T_PIN *cpu_pin,
                T_REGS *reg,
                T_SREGS *sysreg,
                unsigned char *cpuflag
                unsigned char *mpr
                unsigned char *vdc
                unsigned char cpukind)
```

引数 1 ：メモリ領域
引数 2 ：T_PIN *cpu_pin　　バス
引数 3 ：T_REGS reg　　プログラム・レジスタ
引数 4 ：T_SREGS sysreg　　システム・レジスタ
引数 5 ：割り込み完了通知/PC 更新通知
引数 6 ：MPR レジスタ
引数 7 ：VDC
引数 8 ：CPU 種別

第 1 引数の unsigned char *型変数(実際には unsigned char 型配列の先頭アドレスを渡す)でメモリ領域を表現する配列を命令実行関数へ渡すことにより、PC レジスタに格納されたインデックスの示す領域からオペコードを読み取り、命令を実行する。

第 5 引数の割り込み完了通知/PC 更新通知は、関数実行後、通常 PC 値が実行済み命令のアドレスを差しているが、次実行命令のアドレスを差している場合に限り、フラグ PCUPDT(0x80) が有効となる。また、メモリに値が出力された場合、フラグ MEMOUT(0x02)が有効となり、VDC への出力があった場合はフラグ VDCOUT(0x04)が有効となる。

第 8 引数の CPU 種別でエミュレート対象とする CPU を決定する。指定できるマクロは下記の通りである。

MOS6502
W65C02
RP2A03
HuC6280

4. 参考文献

ネット上の数々の文献