

マイクロプロセッサ

1 目的

- (1) 教育用 8 ビット・マイクロプロセッサ：KUE-CHIP2 を用いて，プログラムにおける命令の取り出し，解説，実行の基本サイクルを理解する。
- (2) 種々の課題を通して，計算機のソフト，ハードの両面の理解を実践的に深める。

2 KUE-CHIP2 教育用ボードコンピュータ

2.1 KUE-CHIP2

KUE-CHIP2(Kyoto University Education Chip2) は，計算機の動作原理の理解を目的として，京都大学と京都高度技術研究所 (ASTEM) で開発された教育用 8 ビット・マイクロプロセッサであり，次の特徴を有する。

- (1) プログラムカウンタ，レジスタなどの内容が観察できる。
- (2) クロックフェーズ単位の命令の実行ができる。

2.2 KUE-CHIP2 の構造

図 1 に KUE-CHIP2 のブロック図を示す。破線内が集積回路化されている部分であり，

| | | |
|---------------|---|--------|
| 論理素子 | : | 1597 個 |
| ゲート数 フリップフロップ | : | 62 個 |
| パッド | : | 76 個 |

が，active chip area $5.18 \times 4.38 = 22.65\text{mm}^2$ の中に集積されている。入力データバス DBi，出力データバス DBo，アドレスバス AB の 3 種類のバスの他に，内部のバスやレジスタなどを観察するための観測用バス OB が準備されている。観察バスの内容はボード上の LED や 7 セグメント表示用 LED に表示される。またレジスタ及びメモリの内容は，ボードのスイッチで直接書き込むことができる。

KUE-CHIP2 では演算ユニットとして，ALU が 1 個，演算用レジスタとしてアキュムレータ ACC とインデックスレジスタ IX がそれぞれ準備されている。制御ユニットとしては，プログラムカウンタ PC，命令レジスタ IR，命令デコーダ IDC などが準備されている。

各部の働きは以下の通りである。

2.2.1 演算ユニット ALU

論理演算，算術演算を行う。データの演算の他にアドレスの計算にも用いられる。

2.2.2 レジスタ群

レジスタとして，アキュムレータ ACC，インデックスレジスタ IX，フラグレジスタなどが準備されているが，スタックは準備されていない。

アキュムレータ ACC

演算に利用する 8 ビットのレジスタ。演算のオペランドや演算結果を保持する。

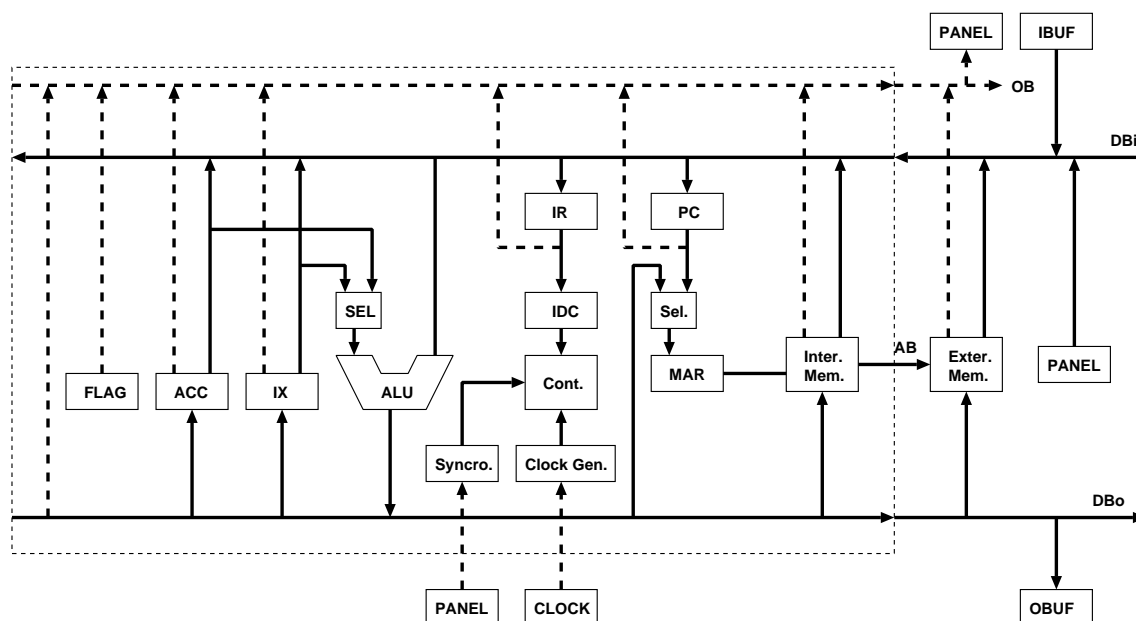


Figure 1: KUE-CHIP2 の構造

インデックスレジスタ **IX**

演算に利用する8ビットのレジスタ。演算のオペランドや演算結果を保持する。また、修飾アドレス指定を行う場合、アドレス修飾に用いられる。

フラグレジスタ

桁上げフラグCF、桁あふれフラグVF、ネガティブフラグNF、ゼロフラグZFの4ビットからなる。演算結果に従いフラグがセット/リセットされる。図2にフラグレジスタを示す。

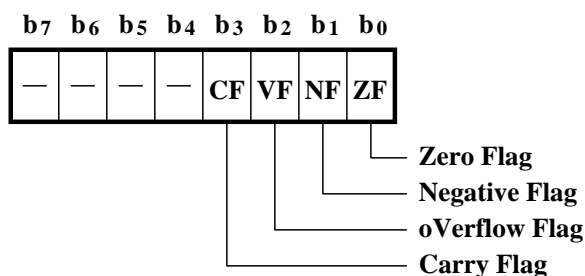


Figure 2: フラグレジスタ

プログラムカウンタ **PC**

次に実行する命令のメモリ上でのアドレスを保持する8ビットのレジスタ。

メモリアドレスレジスタ **MAR**

メモリアクセスのためのアドレスを保持する8ビットのレジスタ。ここに保持されたアドレスの内容がメモリ操作の対象になる。

命令レジスタ **IR**

メモリから読み出した命令を保持する8ビットのレジスタ。

2.2.3 メモリ空間

メモリ空間は512バイトでバイト単位で指定される。0番地～255番地をプログラム領域とよび、プログラムあるいはデータの格納に使用される。一方、256番地～511番地はデータ領域とよ

ばれ、データの格納のみに使用できる。プログラムカウンタが8ビットであるため、プログラムはプログラム領域に置かなければならない。アドレスの最上位ビット(9ビット)は、命令のデコードの後にコントローラにおいて直接生成される。このビットによって、プログラム領域とデータ領域のどちらの領域にアクセスするかが決められる。アドレスの下位8ビットはメモリアドレスレジスタの内容によって指定される。

KUE-CHIP2では、チップ内に512バイトの内部メモリ (Internal Memory)を持っている。チップ外の512バイトの外部メモリ (External Memory) も内部メモリと同等に使用できる。しかし、内部メモリと外部メモリの切り替えは、ボード上のスイッチで行うため、プログラム実行中に両者を切り替えて使用することはできない。図3に KUE-CHIP2のメモリマップを示す。

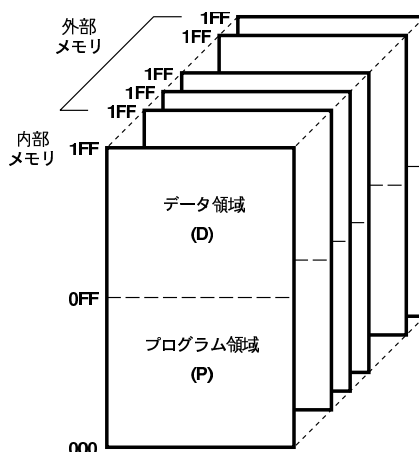


Figure 3: KUE-CHIP2のメモリマップ

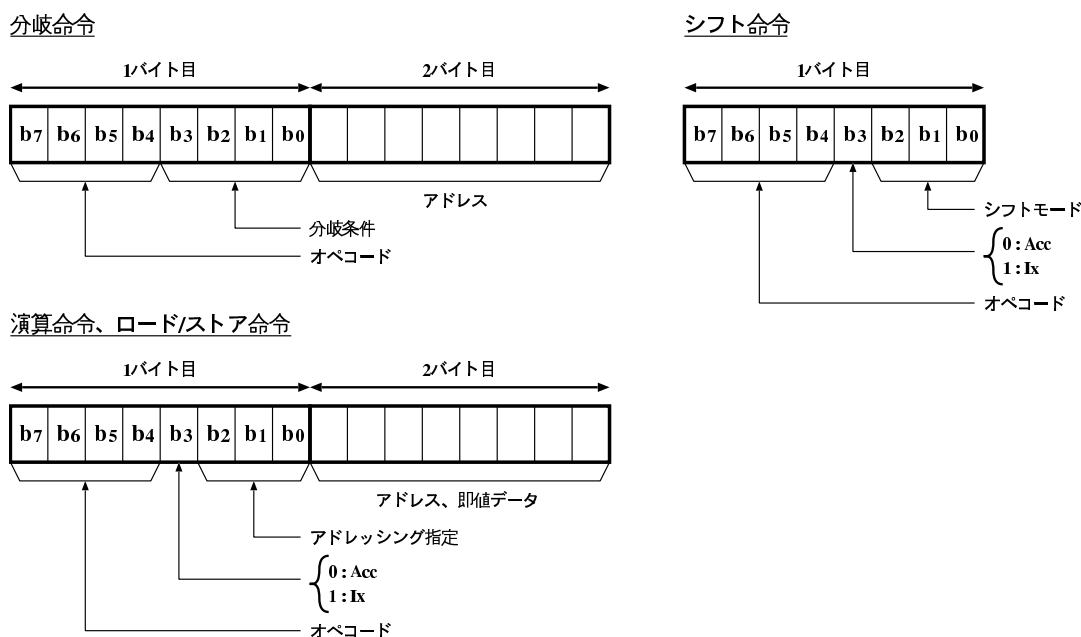


Figure 4: 機械語フォーマット

2.3 命令形式と命令セット

KUE-CHIP2の命令語は1バイトあるいは2バイトで構成される。命令語の1バイト目の上位4ビットあるいは上位5ビット(ビット7からビット4あるいはビット3)で命令の種類を表す。

分岐命令では下位4ビットで分岐条件を表す。シフト命令では下位3ビットがシフトモードを示す。演算命令、ロード/ストア命令では、ビット3が演算用レジスタの指定、下位3ビット(ビット2～ビット0)で第2オペランドのアドレスモードを指定する。これら命令の機械語フォーマットを図4に示す。2バイト命令の2バイト目は、アドレスまたはデータである。データは、2の補数表示で負の数を表し、最上位ビット(第7ビット)が符号に対応する。

付録AにKUE-CHIP2の命令セットを示す。命令は19種類であり、入出力命令2、シフト命令2、算術論理演算命令2、フラグセット命令2、ロード命令、ストア命令、分岐命令、NOP命令、停止命令からなる。命令は可変長で、1バイト命令と2バイト命令からなる。メモリアクセスのアドレス指定は5つのモードがある。すなわち、即値データ、直接アドレス(プログラム領域P、データ領域D)とインデクスレジスタIXによる修飾アドレス(プログラム領域P、データ領域D)の指定方式である。即値データ方式では、2バイト目がそのままデータとなる。直接アドレス方式では、2バイト目がデータが格納されているアドレスを示す。修飾アドレス方式では、2バイト目の内容とインデクスレジスタIXの内容を加算して決まる値が第2オペランドのアドレスとなる。各命令の簡単な動作は表1の通りである。

Table 1: 各命令の動作

| | |
|--------------------------|-------------------------|
| NOP(No OPeration) | 何もしない。 |
| HLT(HaLT) | 停止命令。 |
| OUT(OUTput) | 出力ポートへACCの内容を出力する。 |
| IN(INput) | 入力ポートからACCへデータを取り込む。 |
| RCF(Reset CF) | 桁上げフラグCFをリセットする。 |
| SCF(Set CF) | CFをセットする。 |
| Bcc(Branch cc) | 分岐命令。条件コードccによって分岐。 |
| Ssm(Shift sm) | シフト命令。smはシフトモードの指定。 |
| Rsm(Rotate sm) | 巡回シフト命令。smはシフトモードの指定。 |
| LD(Load) | ロード命令。メモリからデータをレジスタへ移す。 |
| ST(STore) | ストア命令。レジスタの内容をメモリへ格納する。 |
| SBC(SuBtract with Carry) | 減算命令。CFを考慮する。 |
| ADC(ADd with Carry) | 加算命令。CFを考慮する。 |
| SUB(SUBtract) | 減算命令。CFを考慮しない。 |
| ADD(ADD) | 加算命令。CFを考慮しない。 |
| EOR(Exclusive OR) | 各ビット毎の排他的論理和。 |
| OR(OR) | 各ビット毎の論理和。 |
| AND(AND) | 各ビット毎の論理積。 |
| CMP(CoMPare) | 比較命令。 |

2.4 プログラミング及びKUE-CHIP2の動作

KUE-CHIP2はP0～P4の最大5フェーズで1命令を実行をする。KUE-CHIP2には、通常の動作モードのほかに、1命令ごとに動作を停止するシングルインストラクションモードと、1フェーズごとに動作を停止するシングルフェーズモードがあり、これらのモードを利用し各フェーズで回路がどのように動作しているか理解できる。フェーズ動作中、各フェーズ実行によるレジスタ値の変更は、そのフェーズが終了し、次のフェーズに入る前に行われる。

表2にKUE-CHIP2の各命令の実行フェーズ表を示す。フェーズP0の命令のフェッチ及び、フェーズP1に行われる命令レジスタへの命令の転送は、各命令で共通である。

フェーズ：P0では、命令のフェッチが行われる。すなわち、命令の入っているアドレス(プログラムカウンタPCに記憶されている)がメモリアドレスレジスタに送られ、その後、自動的にプログラムカウンタは1加算される。

フェーズ：P1では、メモリアドレスレジスタの内容を番地とするメモリの語がバスDBiを通して命令レジスタに転送される。

Table 2: 命令実行フェーズ

| phase instruction | | P0 | P1 | P2 | P3 | P4 |
|---|----------------------|--|--|--|--|----|
| HLT | | (PC) → MAR PC ++ | (Mem) → IR | HALT | | |
| NOP | | | | NOP | | |
| OUT | | | | (ACC) → OBUF | 0 → OBUF_WE | |
| IN | | | | (IBUF) → ACC 0 → IBUF_RE | 0 → IBUF_FLG_CLR | |
| RCF | | | | 0 → CF | | |
| SCF | | | | 1 → CF | | |
| Bcc | | | | (PC) → MAR PC ++ | STATUS CHECK (Mem) → PC (condition satisfied) | |
| Ssm Rsm | | | | TCF SET SHIFT | NF, ZF, VF, CF SET | |
| LD | ACC IX | | | (A) → B | | |
| | d | | | | | |
| | [d] (d) | | | | | |
| | [IX + d] (IX + d) | | | | | |
| ST | [d] (d) | (PC) → MAR PC ++ | (Mem) → MAR PC ++ | (Mem) → A | (Mem) → A | |
| | [IX + d] (IX + d) | | | (IX) → ALU → MAR (Mem) → ALU → MAR | | |
| SBC ADC SUB ADD EOR OR AND CMP | ACC IX | (PC) → MAR PC ++ | (Mem) → MAR PC ++ | (Mem) → MAR | A → (Mem) | |
| | d | | | (IX) → ALU → MAR (Mem) → ALU → MAR | | |
| | [d] (d) | | | (A) → ALU → A (B) → ALU → A [(CF)] → ALU → A NF, ZF, VF, CF SET | | |
| | [IX + d] (IX + d) | | | | | |
| | | | | | | |
| | | (A) → ALU → A (B) → ALU → A [(CF)] → ALU → A NF, ZF, VF, CF SET | | | | |
| | | (PC) → MAR PC ++ | (A) → ALU → A (B) → ALU → A [(CF)] → ALU → A NF, ZF, VF, CF SET | | | |
| | | | (Mem) → MAR | (A) → ALU → A (B) → ALU → A [(CF)] → ALU → A NF, ZF, VF, CF SET | | |
| | | | (IX) → ALU → MAR (Mem) → ALU → MAR | | | |

フェーズ：P2以降は命令により動作が異なる。例えば、2バイト命令では、2バイト目のアドレスをメモリアドレスレジスタに転送し、ReadあるいはWriteを行なう。その後、プログラムカウンタPCを1増加させる。

実際に簡単な2数の加算プログラムの実行をクロックフェーズ単位で考える。

D1(80h番地)とD2(81h番地)に格納されている2つの符号付き単精度整数(2の補数表現による8ビットデータ)について、その和をANS(82h番地)に格納するプログラムについて考える(アセンブラ文法は付録Aに従った)。

なお、D1、D2の値はあらかじめ、

$$D1 = 3$$

$$D2 = -3$$

に設定されているものとする。また、プログラムは0h番地から実行するようにプログラムカウンタPCが設定されているものとする。

リスト 1

| アドレス | データ | ラベル | 命令 | オペランド | |
|------|-------|------|-----|-----------|-------------|
| | | D1: | EQU | 80h | D1 のアドレスを定義 |
| | | D2: | EQU | 81h | |
| | | ANS: | EQU | 82h | |
| 00: | 64 80 | | LD | ACC,[D1] | |
| 02: | B4 81 | | ADD | ACC,[D2] | |
| 04: | 74 82 | | ST | ACC,[ANS] | |
| 06: | 08 | | HLT | | |
| | | | END | | |
| 80: | 03 | | | | |
| 81: | FD | | | | |

プログラム実行直後 (PC=0) のロード命令 “LD ACC, [D1]” を各フェーズごとにトレースすると次のようになる (表 2, 図 5, 6 を参照)。

P0 (1) PC が示している 0 番地の命令を読み出すために PC の内容 “0” を MAR に転送する。MAR の内容は “0” となる。このあと PC を 1 増加させ, “PC=1” とする。

P1 (2) 命令内容の読み出しが行われる。

MAR の内容で指定された 00h 番地の内容 (64h) が入力バス DBi を通って命令レジスタ IR へ転送される。

P2 (3) 命令デコーダ IDC で IR の内容が解釈される。

上位 4 ビット “0110” はロード命令であることを, 第 3 ビットが “0” であることは対象としているレジスタが ACC であることを, 下位 3 ビット “100” はロードすべきデータのアドレスは直接アドレス方式で記述され, データはプログラム領域にあることを示していると, 解釈される (図 7)。

(4) 次に第 2 オペランドのアドレスを生成するために, 2 バイト目を読み出す。このため, 再び PC の内容 “PC=1” を MAR へ転送し, PC を 1 増加させ “PC=2” とする。MAR の内容は “1” となる。

P3 メモリの読み出しが行われる。

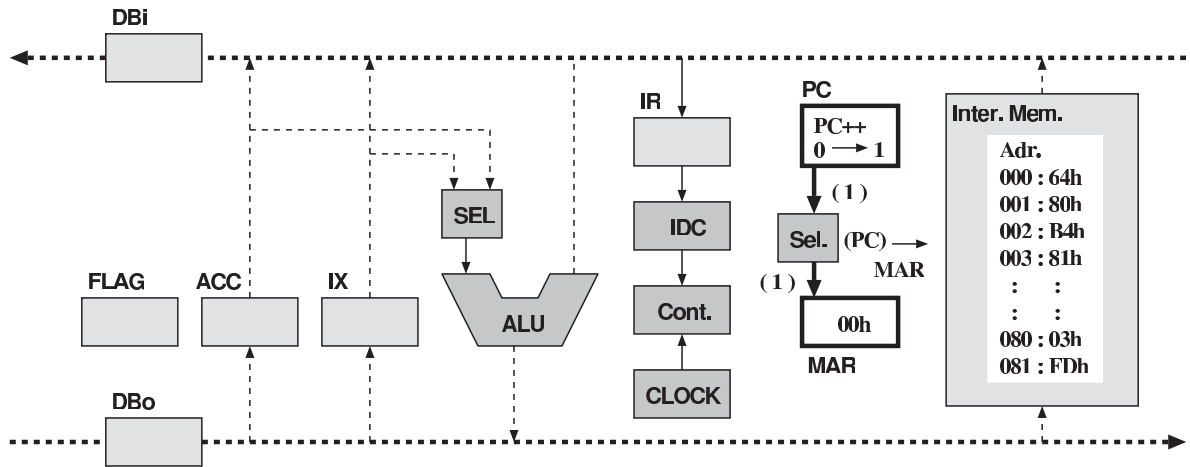
(5) MAR の内容で指された 01h 番地の内容 “80h” が読み出され, DBi → ALU → DBo を介して, MAR へ転送される。MAR の内容は “80h” となる。

P4 第 2 オペランドの読み出しが行われる。

(6) 命令のアドレスモードはプログラム領域を指しているなので, 読み出されるデータのアドレスの最上位ビットは “0” となり, “0 1000 0000” 番地 (80h 番地) となる。

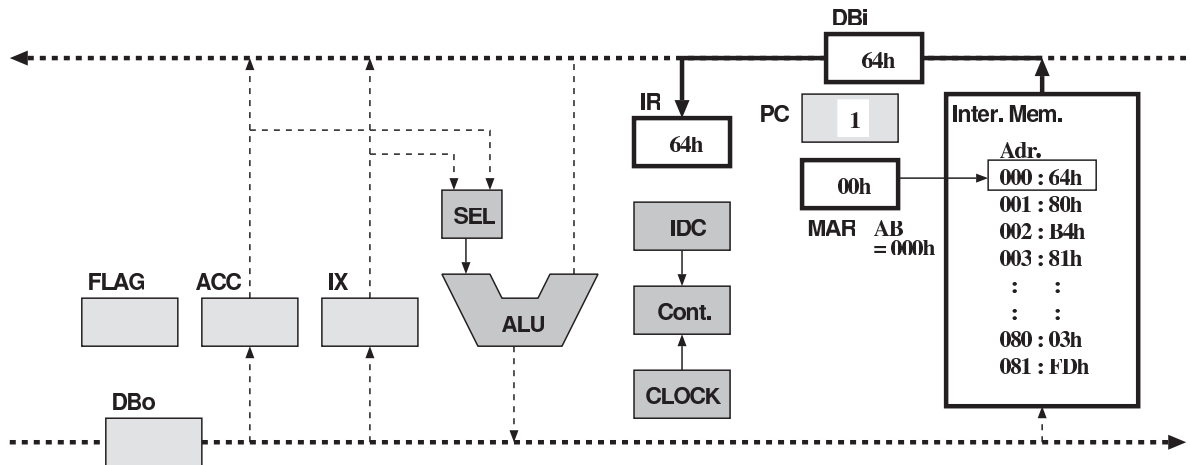
読み出されたデータ (80h 番地の内容) は DBi → ALU → DBo を介して, ACC に格納される。

phase P0 : (PC) \rightarrow MAR , PC++



phase P1 : (Mem) \rightarrow IR

(2) (Mem) \rightarrow IR



phase P2 : (PC) \rightarrow MAR , PC++

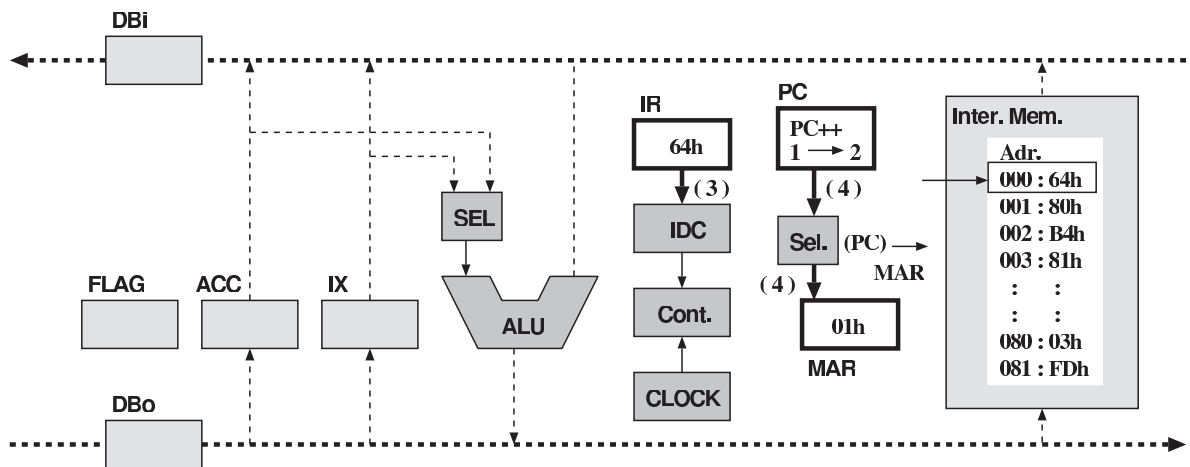
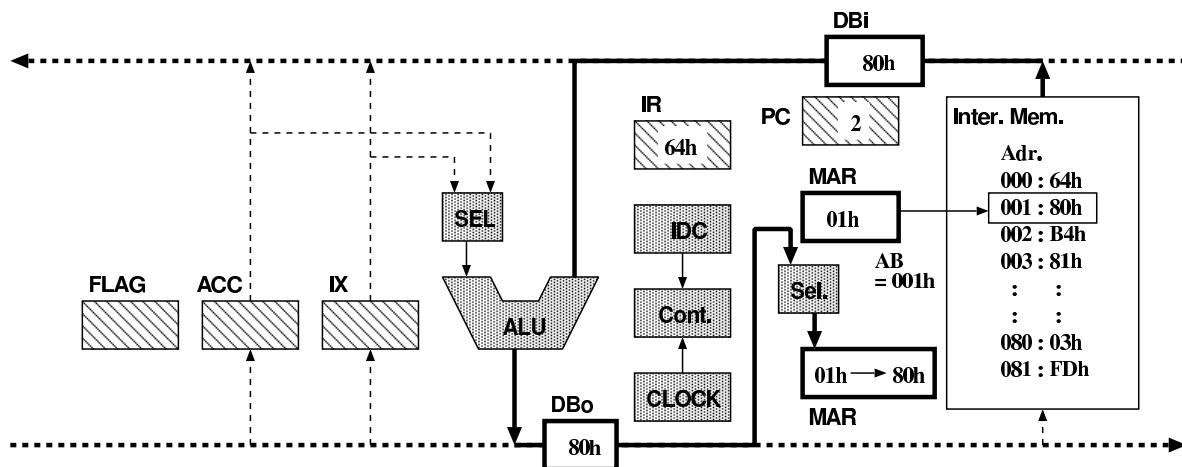


Figure 5: Load 命令の実行 (1)

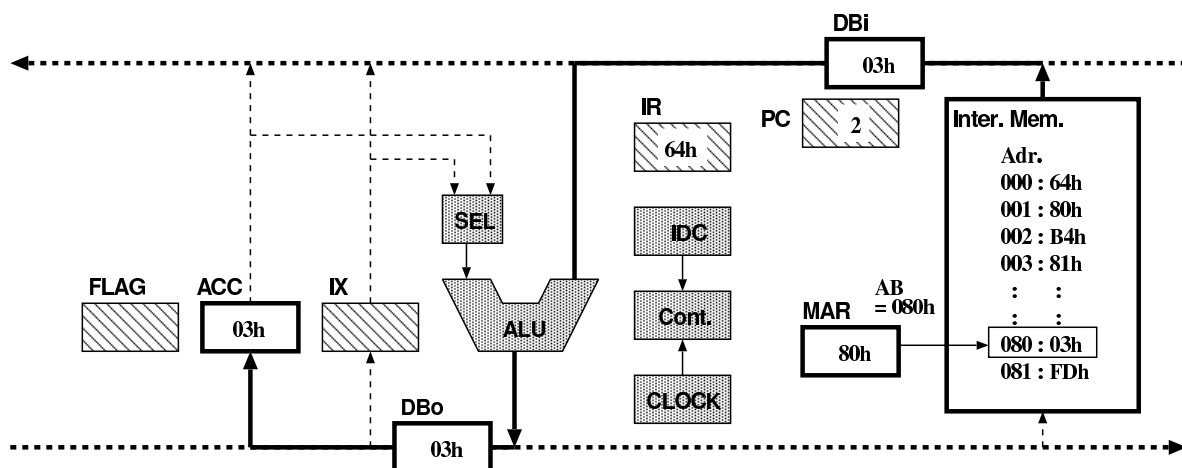
phase P3 : (Mem) → MAR

(5) (Mem) → MAR



phase P4 : (Mem) → A

(6) (Mem) → MAR



- 観測バスで観測可能な部分、フェーズの実行で値が変化する。
- 観測バスで観測可能な部分、フェーズの実行で値が変化しない。
- 観測バスで観測不可能な部分

Figure 6: Load 命令の実行 (2)

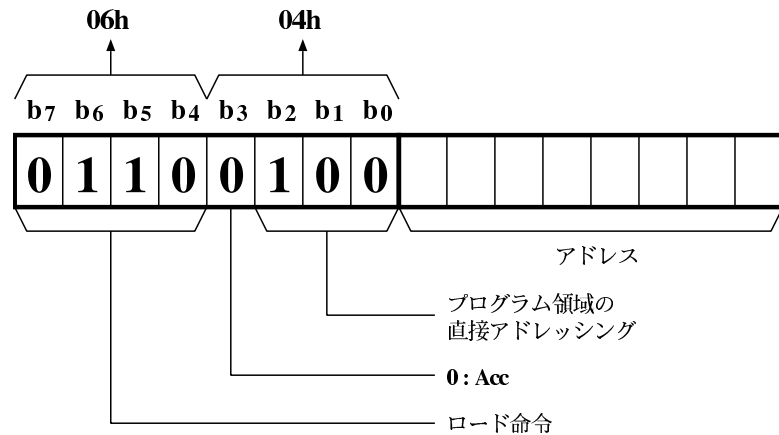


Figure 7: Load 命令のデコード

2.5 プログラムの実行

プログラムの入力の前に KUE-CHIP2 ボードに電源を供給しなければならないが、この時、電源器の出力電圧と極性には十分注意すること。ボードを破壊しないように、丁寧に扱うこと。

2.5.1 ボード上のスイッチおよび表示装置の確認

KUE-CHIP2 ボードの外観を図 8 に示す。ボード上のスイッチおよび表示装置の機能の概要を以下に示す。

(1) 表示装置

- OP : プログラム, 命令の実行中を示す。
- P0 ~ P4 : 実行中のクロックフェーズを示す。
- DATA : 観測バスの値を示す。内容は SEL で選択する。
- ADDRESS : アドレスバスの値を示す。
- IBUF : 入力バッファの状態。
- OBUF : 出力バッファの状態。

(2) スイッチ

- POWER : 電源。
- RESET : ボード全体をリセットする。
- SEL : 観測や書き込みを行なうメモリやレジスタを選択する。
- DATA : 書き込むデータを設定する。
- ADRINC : メモリアドレスレジスタを 1 増加させる。
- ADRDEC : メモリアドレスレジスタを 1 減少させる。
- MEM : 内部メモリ (INT), 外部メモリ (EXT) の切り替え。
- SS : PC の示すアドレスから始まるプログラムを HLT 命令まで実行する。
実行中に押した場合, 現在進行中の命令が終了後, 停止する。
- SI : PC の示すアドレスから 1 命令だけ実行する。
- SP : 現在実行中のクロックフェーズから 1 フェーズだけ実行する。
- CLKFRQ : KUE-CHIP2 の動作するクロック周波数を設定する。

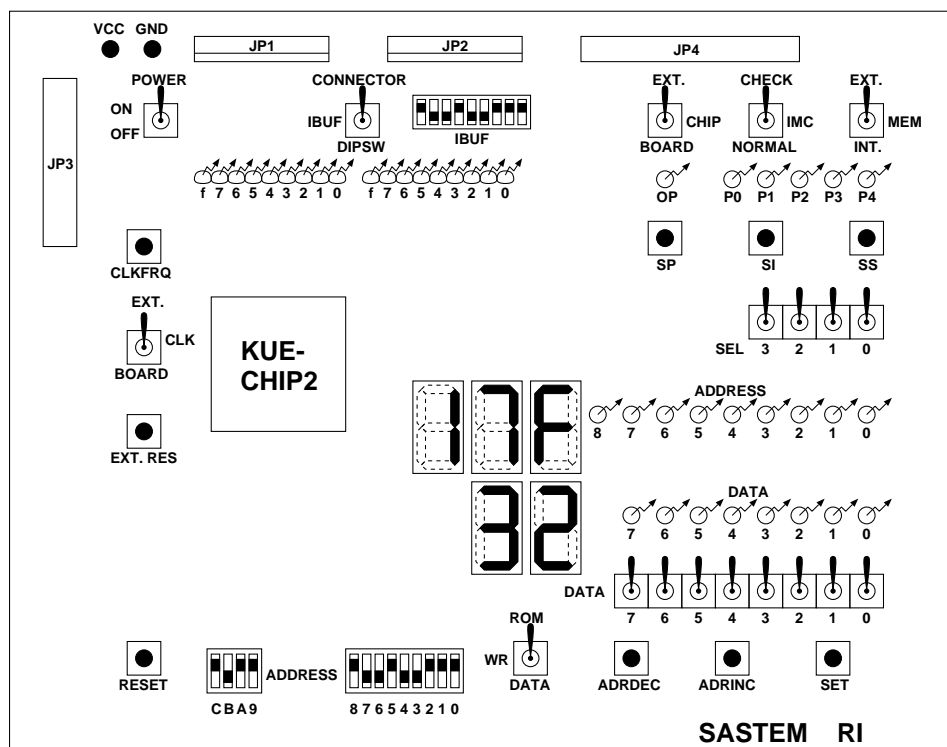


Figure 8: KUE-CHIP2 ボードの外観

(3) その他スイッチの設定

IMC : NORMAL
 CLK : BOARD
 WR : DATA
 IBUF : DIPSW

2.5.2 入力プログラムの例

プログラムの入力が行なわれたかどうかを調べることが容易な例として、出力バッファの状態を示すLEDを右から左へ一つずつ点灯させるプログラムについて、その入力、実行、観察方法を説明する。

プログラムの概要を示す流れ図を図9に示す。

プログラムを00h番地から入力するとして、機械語コードは、

Table 3: SELスイッチの選択

| 3 | 2 | 1 | 0 | 選 択 |
|---|---|---|---|-----------------|
| 0 | 0 | 0 | 0 | MCP プログラム領域メモリ |
| 0 | 0 | 0 | 1 | MCD データ領域メモリ |
| 0 | 0 | 1 | 0 | PC プログラムカウンタ |
| 0 | 0 | 1 | 1 | FLAG フラグレジスタ |
| 0 | 1 | 0 | 0 | ACC アキュムレータ |
| 0 | 1 | 0 | 1 | IX インデックスレジスタ |
| 0 | 1 | 1 | 0 | DBi 入力バス |
| 0 | 1 | 1 | 1 | DBo 出力バス |
| 1 | 0 | 0 | 0 | MAR メモリアドレスレジスタ |
| 1 | 0 | 0 | 1 | IR 命令レジスタ |

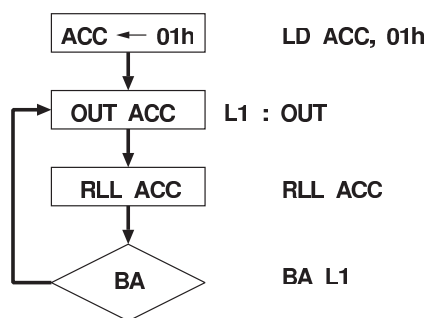


Figure 9: リスト 2 の流れ図

リスト 2

| アドレス | データ | | | | 命令 | オペランド |
|------|------|---------|------|------|-----|----------|
| 00: | 0110 | 001 - | 0000 | 0001 | LD | ACC, 01h |
| 02: | 0001 | 0 - - - | | | OUT | |
| 03: | 0100 | 0111 | | | RLL | ACC |
| 04: | 0011 | 0000 | 0000 | 0010 | BA | 02h |

“-” は0または1のどちらでもよく，ここでは“0”を入れることとすると，実際に入力するコードは，

リスト 3

| アドレス | データ | | | | 命令 | オペランド |
|------|------|------|------|------|-----|----------|
| 00: | 0110 | 0010 | 0000 | 0001 | LD | ACC, 01h |
| 02: | 0001 | 0000 | | | OUT | |
| 03: | 0100 | 0111 | | | RLL | ACC |
| 04: | 0011 | 0000 | 0000 | 0010 | BA | 02h |

となる。

2.5.3 プログラムの入力

次に，リスト 3 のプログラムを入力する方法を説明する。

- (1) 電源を接続し，電源を入れる。
電源器の出力が5Vであることと電源の極性を確認し，電源器，KUE-CHIP ボード上のPOWER スイッチ順でスイッチを入れる。
- (2) それぞれのスイッチを以下のように設定する。

| スイッチ | 位置 |
|--------|--|
| CLKFRQ | 8 : クロックを 100Hz に設定する*。 |
| CLK | BOARD : 内部クロックを使用する。 |
| WR | DATA : 8ビットトグルスイッチよりデータを書き込む。 |
| CHIP | BOARD : ボード上の KUE-CHIP2 を使用する。 |
| IMC | NORMAL : アドレスのバスの下位 9 ビットを KUE-CHIP2 の MAR で与える。 |
| MEM | INT : 内部メモリを使用する。 |

* : 高速なクロックを使用すると，LED が一つずつ点灯するのではなく，8 個が全て同時に点灯しているように見える。

(3) RESET スイッチを押す。

(4) SEL スイッチを “0010” に設定し、PC の内容を確認する。

→ DATA 用 LED が全部消え、7 セグメント表示用 LED が “00” を表示する (PC は “00” である)。

(5) SEL スイッチを “1000” に設定し、MAR の内容を確認する。

→ DATA 用 LED が消え、7 セグメント表示用 LED が “00” を表示する。

(6) SEL スイッチを “0000” にする。

→ ADDRESS 用 LED は変化せず、DATA 用 LED はメモリの 00h 番地の値 (不定) を表示する。

(7) DATA スイッチをメモリの 00h 番地に書き込みたい値である “0110 0010” にする。

→ 表示は変わらない。

(8) SET スイッチを押す。

→ DATA 用 LED が “0110 0010” を示す (メモリに値が書き込まれた)。

(9) ADRINC スイッチを押す。

→ ADDRESS 用 LED が “01h” を示す (MAR がインクリメントされた)。

(10) 7 から 9 までを繰り返し、リスト 3 のプログラムを全て入力する。

2.5.4 プログラムの実行

普通の動作

(RESET スイッチを押してから) SS スイッチを押す。

→ 連続動作しているので、DBo の状態を示す LED は右から左へ一つずつ点灯する。

プログラムが動作しなかった場合

メモリの確認

(1) プログラムが動作中 (黄色の OP 用 LED が点灯し、P0~P4 用 LED が順次点灯) であれば、SS スイッチを押してプログラムの実行を停止する。

(2) RESET スイッチを押す。RESET スイッチにより、ACC, PC, MAR などレジスタはリセットされるが、メモリの内容はリセットされない。SEL スイッチを使い “PC=00h”, “MAR=00h” になっているか確認する。

(3) ADRINC スイッチ, ADRDEC スイッチを用い、各番地の内容を確認する。

(4) 誤りが見つかった場合、その番地の内容を書き換える。

(5) プログラムの終わり (05h 番地) まで、3, 4 を繰り返す。

プログラムのトレース

(1) SS スイッチを押し、プログラムの実行を停止する。

(2) RESET スイッチを押し、PC, MAR をリセットする。

(3) SI スイッチを押し、1 命令だけを実行する。

→ 正常な実行であれば、7 セグメント表示用 LED は
(00h → 01h →) 02h → 03h → 05h
を繰り返す。

(4) このとき、ACC の内容を確認する。

3 実験課題

3.1 加算プログラムのトレース

リスト 1 の加算プログラムをクロックフェーズ単位で実行し、各クロックフェーズ終了時の FLAG, ACC, IR, PC など観測可能なレジスタをトレースせよ。ただし、計算式には下記の (1) から (6) を用い、(1) では観測用バス OB で観測可能な全ての項目についてトレースせよ。また、(2) から (6) では ADD 命令実行中の FLAG についてトレースせよ。

さらに、(2) から (6) については、ADD 命令を ADC 命令に置き換えた場合の FLAG についてもトレースせよ。

なお、値が正しく入力されていることを確かめるため、トレースの前に一度実行して正しい加算結果が得られていることを確認せよ。また、その加算結果も記録しておくこと。

- | | |
|-------------------|-----------------------|
| (1) $2 + 3 = ?$ | (4) $2 + (-3) = ?$ |
| (2) $126 + 1 = ?$ | (5) $-127 + (-1) = ?$ |
| (3) $126 + 2 = ?$ | (6) $-127 + (-2) = ?$ |

3.2 乗算プログラムの作成

符号なし 2 バイト精度の 2 数の積を求めるプログラムを作成し、KUE-CHIP2 ボード上でその動作を確認せよ。ただし、2 バイト精度整数のメモリ上でのデータ記憶形式は図 10(a) の通りである。例えば、被乗数は 80h 及び 81h 番地に、乗数は 82h 及び 83h 番地に格納されているものとする。また、演算結果は 84h 及び 85h 番地へ格納するものとする。

また、余力のある者は、2 の補数を用いて表わされた符号付き 2 バイト精度の 2 数の積を求めるプログラムを作成してみよ。例えば “-300” の 2 の補数表現 “FED4h” は図 10(b) の様に格納されている。

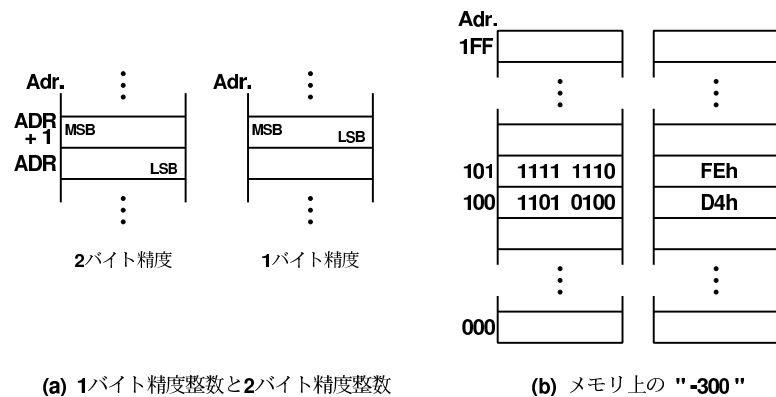


Figure 10: 2 バイト精度整数

3.3 メロディーの出力(付録B参照)

- (1) KUE-CHIP ボードを用いて 440Hz (「ラ」の音) の方形波をスピーカに出力させよ。ただし、周波数の誤差は $\pm 1\%$ 以下とする。
 - (a) KUE-CHIP ボードのクロックの周期をオシロスコープで確認せよ。KUE-CHIP ボードのクロックはスイッチで可変であるので、スイッチ CLKFRQ を操作して 8 番から 0 番まで全てのクロックを確認せよ。クロック周期の測定は、CLK スイッチを中立にし、コネクタ JP3 (3 番ピン) からクロック信号を取得せよ。
 - (b) (a) で測定した周波数を用い、440Hz の方形波を出力するプログラムを作成せよ。なお、付録 B にあるプログラムを参考にする場合には、最適な T_0 と a, b (つまり、誤差が最も小さくなるもの) を考えること。
 - (c) 作成したプログラムを実行し、440Hz ($\pm 1\%$) の周波数がスピーカに出力されていることを確認せよ。また、オシロスコープで出力波形を観測せよ。
- (2) KUE-CHIP2 ボードを用い、メロディーを出力するプログラムを作成せよ。ただし、休符や連続した同じ高さの音にも対応できるプログラムにすること (すなわち、リスト 5 のプログラムそのままでは不可とする)。そして、実際に簡単なメロディーをスピーカに出力させよ。ただし、1 曲を全部出力する必要はなく、1 部分だけでよい。楽譜は各自で用意すること。

4 検討事項

- (1) “3.1 加算プログラムのトレース” の結果について、フェーズ単位の実行中の各レジスタの値の変化をまとめよ。2.4 節を参考に、“図 1 KUE-CHIP2 の構造” なども用いながら文章で分かりやすく説明すること。また、各 FLAG がどのような場合に变化するのか考察すること。
- (2) “3.2 乗算プログラムの作成” について、そのアルゴリズムを他の学生のプログラムと比較し、コードの大きさ、実行速度の観点からまとめよ。さらに、作成したプログラム、および表 2 を参考に、計算を完了するまでのステップ (クロック) 数から計算時間の理論値を求めて比較せよ。なお、実験報告書にはその結果だけでなく、計算過程も示すこと。
- (3) “3.3 メロディーの出力” について、以下の事項について考察せよ。
 - (a) $\pm 1\%$ の精度であった確認をどのように行なったか。また、他の方法も考えられるか。
 - (b) 精度をより上げるためには、どのような対策を行なうことが考えられるか。プログラムの改良により KUE-CHIP2 ボードだけで対処する場合と、その他の機器を KUE-CHIP2 ボードに接続する場合で考えよ。
 - (c) 各自のメロディーのデータ表現の特徴を述べよ。他にどのようなデータ表現が考えられるか。人間にとって分かり易いデータ表現、あるいは音階であることをうまく利用したデータ表現、という観点から考えてみよ。
 - (d) 各自のメロディーを出力する方法は、他のプロセッサ (例えば、Z80 や 80286) に対しても流用することができるか否かを考察せよ。ただし、他のプロセッサの実行命令フェーズを調べ、その理由も添えること。もし流用できないならば、他のプロセッサでは、どのような点を考慮に入れて、メロディー (方形波) を出す必要があるかを述べよ。
- (4) 余力のある者は、現在、自分がよく使用している CPU (または、有名な CPU) について、レジスタ、命令セット、メモリ空間の特徴を述べながら、そのアーキテクチャについてまとめよ。また、乗算命令がどのように実行されているかまとめよ。

References

- [1] 神原弘之 他: “KUE-CHIP2 教育用ボードリファレンスマニュアル Version 1.10”(1993)
- [2] 神原弘之 他: “KUE-CHIP2 設計ドキュメント Version1.10”(1993)
- [3] “理科年表”, 丸善 (1993)
- [4] 神原弘之, 安浦寛人: “計算機教育用マイクロコンピュータの開発とその応用 – 集積回路技術を利用した情報工学実験 –, 情報処理, **Vol.33**, No.2, pp.118–127(1992)
- [5] 成田福雄: “Z80 演算サブルーチン・ライブラリ”, pp.17–40, 61–80, 工学社 (1986)
- [6] 西沢 昭: “Z80 上級プログラミング”, pp.32–69, 176–218, CQ 出版 (1984)
- [7] 稲葉 保: “精選アナログ実用回路集”, pp.206–215, CQ 出版 (1989)
- [8] 横山直隆: “パソコン・インターフェースの製作自習”, pp.242–265, 技術評論社 (1986)
- [9] “特集 マイコン周辺回路 完全マスタ”, トランジスタ技術, **Vol.28**, No.5, pp.338–466(1989)
- [10] 阿部英志: “8086 CPU のプログラミング・モデル”, ブートストラップ, **No.2**, pp.16–67, CQ 出版 (1992)
- [11] 丸岡崇 : “ロジック IC を使ったコンピュータ設計入門”, トランジスタ技術, **Vol.29**, No.9, pp.309–327(1992)
- [12] “特集 マイコン応用設計マニュアル”, トランジスタ技術, **Vol.26**, No.6, pp.404–486(1989)

A KUE-CHIP2 の命令仕様

A.1 アセンブラ文法

KUE-CHIP2 のアドレスモード

| | |
|----------|------------------------|
| ACC | アキュムレータ |
| IX | インデックスレジスタ |
| d | 即値データ |
| (d) | 直接アドレス (データ領域) |
| [d] | 直接アドレス (プログラム領域) |
| (IX + d) | インデックス修飾アドレス (データ領域) |
| [IX + d] | インデックス修飾アドレス (プログラム領域) |

KUE-CHIP2 論理アドレスモード

| | |
|-------|--------------|
| {ea} | 全てのアドレスモード |
| {reg} | レジスタアドレスモード |
| {imm} | 即値データモード |
| {ma} | メモリ参照アドレスモード |

アドレスモード対応表

| | ACC | IX | d | (d) | [d] | (IX+d) | [IX+d] |
|-------|-----|----|---|-----|-----|--------|--------|
| {ea} | ○ | ○ | ○ | ○ | ○ | ○ | ○ |
| {reg} | ○ | ○ | × | × | × | × | × |
| {imm} | × | × | ○ | × | × | × | × |
| {ma} | × | × | × | ○ | ○ | ○ | ○ |

KUE-CHIP2 命令表

| 略記号 | アドレスモード | 命令機能 |
|------------|------------|-----------------------------|
| HLT | | Halt |
| NOP | | No Operation |
| IN | | INput |
| OUT | | OUTput |
| SCF | | Set Carry Flag |
| RCF | | Reset Carry Flag |
| LD | {reg},{ea} | LoaD |
| ST | {reg},{ma} | STore |
| ADD | {reg},{ea} | ADD |
| ADC | {reg},{ea} | ADd with Carry |
| SUB | {reg},{ea} | SUBtract |
| SBC | {reg},{ea} | SuBtract with Carry |
| CMP | {reg},{ea} | CoMPare |
| AND | {reg},{ea} | AND |
| OR | {reg},{ea} | OR |
| EOR | {reg},{ea} | Exclusive OR |
| <i>Ssm</i> | {reg} | Shift |
| <i>Rsm</i> | {reg} | Rotate |
| RA | | Right Arithmetically |
| LA | | Left Arithmetically |
| RL | | Right Logically |
| LL | | Left Logically |
| Bcc | {imm} | Branch |
| A | | Always |
| VF | | on oVerFlow |
| NZ | | on Not Zero |
| Z | | on Zero |
| ZP | | on Zero or Positive |
| N | | on Negative |
| P | | on Positive |
| ZN | | on Zero or Negative |
| NI | | on No Input |
| NO | | on No Output |
| NC | | on No Carry |
| C | | on Carry |
| GE | | on Greater than or Equal to |
| LT | | on Less Than |
| GT | | on Greater Than |
| LE | | on Less than or Equal to |

A.2 命令セット

| 記号 | 命令コード (1 バイト目, 2 バイト目) | | | | | | | | 命令機能の概略 | | |
|-----|------------------------|---|---|---|---|---|---|---|----------------|--------------------|-------------------------|
| NOP | 0 | 0 | 0 | 0 | 0 | — | — | × | No OPERATION | | |
| HLT | 0 | 0 | 0 | 0 | 1 | — | — | × | Halt | 停止 | |
| | 0 | 1 | 0 | 1 | — | — | — | × | | 未使用 (HLT) | |
| OUT | 0 | 0 | 0 | 1 | 0 | — | — | × | OUTput | (ACC) → OBUF | |
| IN | 0 | 0 | 0 | 1 | 1 | — | — | × | INput | (IBUF) → ACC | |
| RCF | 0 | 0 | 1 | 0 | 0 | — | — | × | Reset CF | 0 → CF | |
| SCF | 0 | 0 | 1 | 0 | 1 | — | — | × | Set CF | 1 → CF | |
| Bcc | 0 | 0 | 1 | 1 | | c | c | ◎ | Branch cc | 条件が成立すれば B' → PC | |
| Ssm | 0 | 1 | 0 | 0 | A | 0 | s | m | × | Shift sm | (A) → shift, rotate → A |
| Rsm | 0 | 1 | 0 | 0 | A | 1 | s | m | × | Rotate sm | はみ出したビット → CF |
| LD | 0 | 1 | 1 | 0 | A | | B | ○ | LoaD | (B) → A | |
| ST | 0 | 1 | 1 | 1 | A | | B | ◎ | STore | (A) → B | |
| SBC | 1 | 0 | 0 | 0 | A | | B | ○ | SuB with Carry | (A) − (B) − CF → A | |
| ADC | 1 | 0 | 0 | 1 | A | | B | ○ | ADd with Carry | (A) + (B) + CF → A | |
| SUB | 1 | 0 | 1 | 0 | A | | B | ○ | SUBtract | (A) − (B) → A | |
| ADD | 1 | 0 | 1 | 1 | A | | B | ○ | ADD | (A) + (B) → A | |
| EOR | 1 | 1 | 0 | 0 | A | | B | ○ | Exclusive OR | (A) ⊕ (B) → A | |
| OR | 1 | 1 | 0 | 1 | A | | B | ○ | OR | (A) ∨ (B) → A | |
| AND | 1 | 1 | 1 | 0 | A | | B | ○ | AND | (A) ∧ (B) → A | |
| CMP | 1 | 1 | 1 | 1 | A | | B | ○ | CoMPare | (A) − (B) | |

cc : Condition Code

| | | | | | | |
|----|---|---|---|---|-----------------------------|---|
| A | 0 | 0 | 0 | 0 | Always | 常に成立 |
| VF | 1 | 0 | 0 | 0 | on oVerFlow | 桁あふれ $VF = 1$ |
| NZ | 0 | 0 | 0 | 1 | on Not Zero | $\neq 0$ $ZF = 0$ |
| Z | 1 | 0 | 0 | 1 | on Zero | $= 0$ $ZF = 1$ |
| ZP | 0 | 0 | 1 | 0 | on Zero or Positive | ≥ 0 $NF = 0$ |
| N | 1 | 0 | 1 | 0 | on Negative | < 0 $NF = 1$ |
| P | 0 | 0 | 1 | 1 | on Positive | > 0 $(NF \vee ZF) = 0$ |
| ZN | 1 | 0 | 1 | 1 | on Zero or Negative | ≤ 0 $(NF \vee ZF) = 1$ |
| NI | 0 | 1 | 0 | 0 | on No Input | $IBUF_FLG_IN = 0$ |
| NO | 1 | 1 | 0 | 0 | on No Output | $OBUF_FLG_IN = 0$ |
| NC | 0 | 1 | 0 | 1 | on Not Carry | $CF = 0$ |
| C | 1 | 1 | 0 | 1 | on Carry | $CF = 1$ |
| GE | 0 | 1 | 1 | 0 | on Greater than or Equal to | ≥ 0 $(VF \oplus NF) = 0$ |
| LT | 1 | 1 | 1 | 0 | on Less than | < 0 $(VF \oplus NF) = 1$ |
| GT | 0 | 1 | 1 | 1 | on Greater than | > 0 $((VF \oplus NF) \vee ZF) = 0$ |
| LE | 1 | 1 | 1 | 1 | on Less than or Equal to | ≤ 0 $((VF \oplus NF) \vee ZF) = 1$ |

sm : Shift Mode

| | | | |
|----|---|---|----------------------|
| RA | 0 | 0 | Right Arithmetically |
| LA | 0 | 1 | Left Arithmetically |
| RL | 1 | 0 | Right Logically |
| LL | 1 | 1 | Left Logically |

$A = 0$: ACC

$A = 1$: IX

B' (2 バイト目)

× : 不用

○ : 不用 or 必要

◎ : 必要

$B = 000$: ACC

$B = 001$: IX

$B = 01-$: 即値 (B' : データ)

$B = 100$: 直接 (P) (B' : アドレス)

$B = 101$: 直接 (D) (B' : アドレス)

$B = 110$: 修飾 (P) ($B' + (IX)$: アドレス)

$B = 111$: 修飾 (D) ($B' + (IX)$: アドレス)

A.3 Shift / Rotate の命令の機能

| 略記号 | MSB | LSB |
|-----|-----|-----|
| SRA | | |
| SLA | | |
| SRL | | |
| SLL | | |
| RRA | | |
| RLA | | |
| RRL | | |
| RLL | | |

B メロディの出力

B.1 音の出力

スピーカへの音の出力は、KUE-CHIP ボードからスピーカへ方形波を出力することによって実現する。スピーカの回路構成を図 11 に示す。ここではスピーカは D/A コンバータに接続するものとし、KUE-CHIP ボードからは 00h と FFh を交互に出力する。

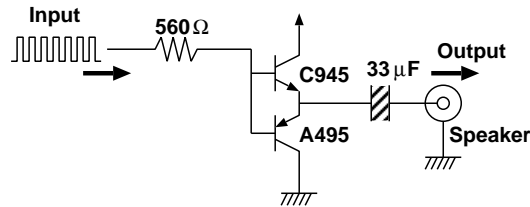


Figure 11: スピーカの回路構成

方形波を出力するプログラムとしてリスト 4 を考える。このプログラムは、コネクタ JP1 の全ビットに方形波を出力する。プログラムより 003 番地から 00B 番地まで、コネクタ JP1 の出力は on になる。また、000 番地と 001 番地から 002 番地および 00C 番地から 012 番地まで、コネクタ JP1 の出力は off となる。このとき 002 番地から 00A 番地まで実行に要する時間 T_a は、

$$T_a = (12 + 8a) \cdot T_0$$

となる。ここで、 T_0 はクロック周期である。000 番地と 001 番地および 00B 番地から 012 番地までの実行に要する時間 T_b は、

$$T_b = (16 + 8b) \cdot T_0$$

となる。スピーカに出力される方形波の周期 T は T_a と T_b の和で、

$$T = T_a + T_b$$

リスト 4

| アドレス | データ | ラベル | 命令 | オペランド | フェーズ数 |
|-------|-------------|-----|-----|---------------|-------|
| 000 : | 62 FF | L0 | LD | ACC, FFh | 4 |
| 002 : | 10 | | OUT | | 4 |
| 003 : | 62 <i>a</i> | | LD | ACC, <i>a</i> | 4 |
| 005 : | A2 01 | L1 | SUB | ACC, 01h | 4 |
| 007 : | 31 05 | | BNZ | L1 | 4 |
| 009 : | 62 00 | | LD | ACC, 00h | 4 |
| 00B : | 10 | | OUT | | 4 |
| 00C : | 62 <i>b</i> | | LD | ACC, <i>b</i> | 4 |
| 00E : | A2 01 | L2 | SUB | ACC, 01h | 4 |
| 010 : | 31 0E | | BNZ | L2 | 4 |
| 012 : | 30 00 | | BA | L0 | 4 |

である（図 12）。方形波の周期を調整するためには、*a*, *b* の値を変更したり、ループ中に NOP などの無意味な命令列を挿入すれば良い。

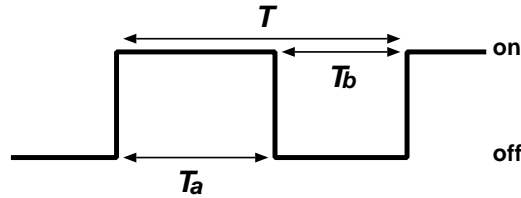


Figure 12: 方形波出力

なお、周期 T が可聴領域の値 (周波数 $(1/T)$ で約 20 ~ 20000 Hz) なら、スピーカに出力した時に“音”として観測できる。ちなみに、リスト 4（およびリスト 5）のプログラムでは、可聴領域を超える高周波・低周波を実現できないため、周波数の調整だけでは無音状態を作ることはいできない。

B.2 メロディー出力プログラム例

リスト 5 にメロディー出力プログラム例を示す。これはドレミファソラシドを 1 秒ずつ出力するプログラムの一部である（データ領域は省略してある）。

各音のデータは 3 バイトからなり、プログラム中の n_1 , n_2 , n_3 に対応する。このときの出力周波数 f および音の長さ T_{beep} は、

$$f = \frac{1}{2 \cdot (8 \cdot n_1 + 41) \cdot T_0}$$

$$T_{\text{beep}} = (((8 \cdot n_1 + 41) \cdot n_2 + 28) \cdot n_3 + 39) \cdot T_0$$

となる。音の高さを調節するパラメータは n_1 である。音の長さを調節するパラメータは n_2 と n_3 であり、 n_2 だけではごく短い音しか表現できないため、 n_3 を導入している。また、dptr1 と dptr2 はそれぞれデータ領域の上限と下限を表す。image は出力するビットイメージ (00h または FFh) であり、方形波を出力するため、プログラム中ではビットイメージを参照した後、反転した値を格納している。さらに、dptr は現在参照しているデータのアドレス、 n_2 および n_3 はそれぞれ上述した n_2 と n_3 の値を格納する領域である。それぞれ、データ領域に格納領域を用意すること（データ領域のアドレスを指定すること、各自のメロディーに合わせて下線部のアドレスを設定すること）。

なお、各音階の周波数は表 13 を参考にせよ。1 オクターブ高い音は周波数が 2 倍になり、低い音は 1/2 になる。また、リスト 5 のプログラムに変更を加えた場合には、上記のパラメータ計算式も変更する必要がある。

Table 4: 音階の周波数 [Hz]

| | | | |
|-----|--------|----|---------------|
| ド | 261.63 | ソ | 392.00 |
| ド# | 277.18 | ソ# | 415.30 |
| レ | 293.66 | ラ | 440.00 |
| レ# | 311.13 | ラ# | 466.16 |
| ミ | 329.63 | シ | 493.88 |
| ファ | 349.23 | ド | 523.25 |
| ファ# | 369.99 | | |

リスト 5

| | | | | |
|-------|--------------|-------|-----|--------------|
| 000 : | 62 00 | | LD | ACC, dptr1 |
| 002 : | 75 <u>1A</u> | | ST | ACC, (dptr) |
| 004 : | 65 <u>1A</u> | L0 : | LD | ACC, (dptr) |
| 006 : | 68 | | LD | IX, ACC |
| 007 : | B2 03 | | ADD | ACC, 03h |
| 009 : | 75 <u>1A</u> | | ST | ACC, (dptr) |
| 00B : | A2 <u>18</u> | | SUB | ACC, dptr2 |
| 00D : | 31 13 | | BNZ | L1 |
| 00F : | 62 00 | | LD | ACC, dptr1 |
| 011 : | 75 <u>1A</u> | | ST | ACC, (dptr) |
| 013 : | 67 02 | L1 : | LD | ACC, (IX+2) |
| 015 : | 75 <u>1C</u> | | ST | ACC, (n3) |
| 017 : | 67 01 | Ln3 : | LD | ACC, (IX+1) |
| 019 : | 75 <u>1B</u> | | ST | ACC, (n2) |
| 01B : | 65 <u>19</u> | Ln2 : | LD | ACC, (image) |
| 01D : | 10 | | OUT | |
| 01E : | C2 FF | | EOR | ACC, FFh |
| 020 : | 75 <u>19</u> | | ST | ACC, (image) |
| 022 : | 67 00 | | LD | ACC, (IX+0) |
| 024 : | A2 01 | Ln1 : | SUB | ACC, 01h |
| 026 : | 31 24 | | BNZ | Ln1 |
| 028 : | 65 <u>1B</u> | | LD | ACC, (n2) |
| 02A : | A2 01 | | SUB | ACC, 01h |
| 02C : | 75 <u>1B</u> | | ST | ACC, (n2) |
| 02E : | 31 1B | | BNZ | Ln2 |
| 030 : | 65 <u>1C</u> | | LD | ACC, (n3) |
| 032 : | A2 01 | | SUB | ACC, 01h |
| 034 : | 75 <u>1C</u> | | ST | ACC, (n3) |
| 036 : | 31 17 | | BNZ | Ln3 |
| 038 : | 30 04 | | BA | L0 |

Microprocessors

1 Objective

- (1) Using an educational 8-bit microprocessor (KUE-CHIP2) understand the basic cycle of retrieving, decoding, and executing instructions in a program.
- (2) Through various tasks, practically enhance understanding of both the software and hardware aspects of computers.

2 Educational Board Computer: KUE-CHIP2

2.1 KUE-CHIP2

KUE-CHIP2 (Kyoto University Education Chip 2) is an educational 8-bit microprocessor developed in Kyoto University and the Advanced Science, Technology & Management Research Institute of KYOTO (ASTEM). It is designed to enable an understanding of the operational principles of computers, and has the following characteristics.

- (1) The interior of components, such as the program counters and registers, can be observed.
- (2) It can execute instructions in clock-phase units.

2.2 KUE-CHIP2 Architecture

Figure 1 shows a block diagram of KUE-CHIP2. The area inside the dashed rectangle is the integrated-circuit section and the following elements integrated into an active chip area of 22.65 mm^2 ($5.18 \text{ mm} \times 4.38 \text{ mm}$):

| Number of the gates | | |
|---------------------|---|-------|
| Logic elements | : | 1,597 |
| Flip-flops | : | 62 |
| Pads | : | 76 |

In addition to three types of bus (an input data bus DBi, an output data bus DBo, and an address bus AB) there is also an observation bus OB for observing internal buses, registers, etc. The content of the observation bus is displayed by LEDs on the board and by seven-segment LED displays. The content of registers and memories can be set directly by switches on the board.

In the KUE-CHIP2, there is one arithmetic logic unit (ALU) and two arithmetic registers: one accumulator (ACC) and one index register (IX). It also includes a program counter (PC), a instruction register (IR), and an instruction decoder (IDC) as control units. The components work as follows.

2.2.1 Arithmetic Logic Unit (ALU)

This unit performs logical and arithmetic operations. In addition to data operations, it is also used for the calculation of address.

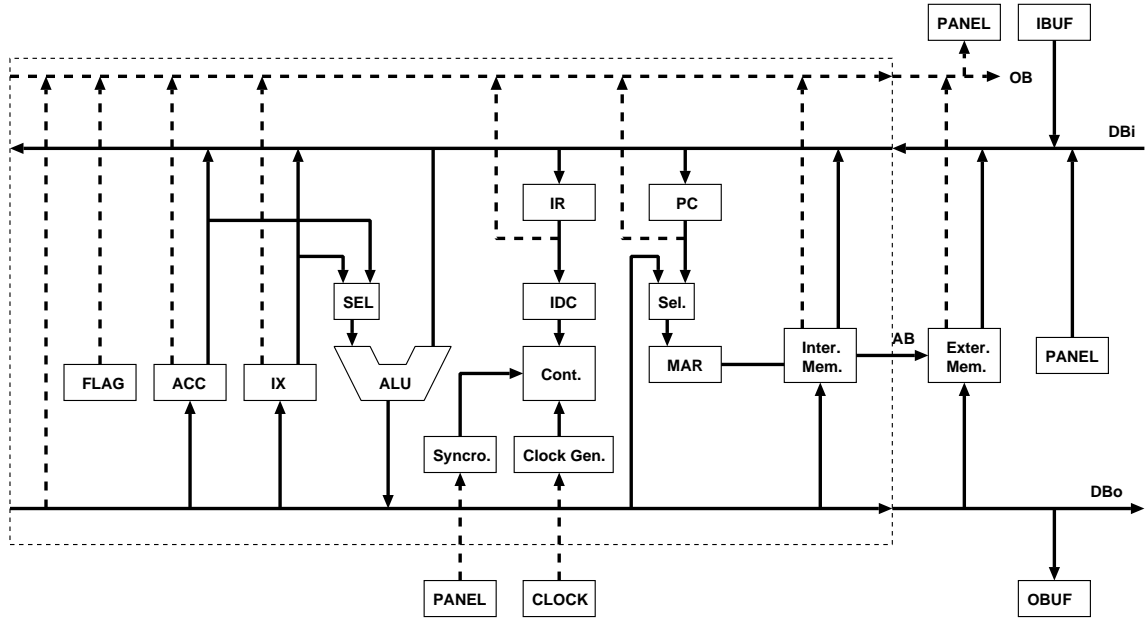


Figure 1: KUE-CHIP2 Architecture

2.2.2 Registers

Registers include the accumulator (ACC), the index register (IX), and a FLAGS register, among others. There is no stack.

Accumulator (ACC)

This 8-bit register is used for arithmetic operations. It stores arithmetic operands and results.

Index Register (IX)

This 8-bit register is used for arithmetic operations. It stores arithmetic operands and results, and is also used for the index when indexed addressing.

FLAGS Register

This register comprises four bits: the carry flag (CF), overflow flag (VF), negative flag (NF), and zero flag (ZF). The flags are set or reset in accordance with the arithmetic result. Figure 2 shows the flag register.

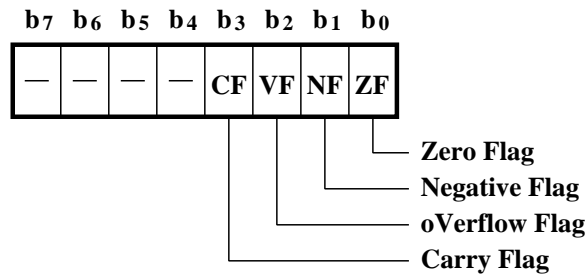


Figure 2: Flags register

Program Counter (PC)

This 8-bit register stores the address of the instruction in the memory space to be executed next.

Memory Address Register (MAR)

This 8-bit register stores an address for memory access. The address content stored here is the target of memory operations.

Instruction Register (IR)

This 8-bit register holds an instruction loaded from the memory.

2.2.3 Memory Space

The memory space is addressed in bytes and consists of 512 bytes. Memory locations 0 to 255 are called the program area and used for storing programs or data. Meanwhile, locations 256 to 511 are called the data area, and can be used only for storing data. Since the program counter is 8 bits long, the programs are necessarily placed in the program area. The most significant bit of the address (9-bit) is generated directly by the controller after decoding the instruction. This bit determines which of the program or data areas are to be accessed. The lower 8 bits of the address are addressed by the content of the memory address register.

KUE-CHIP2 contains 512 bytes of internal memory, inside the chip. An external 512-byte memory, outside the chip, can also be used in the same way as the internal memory. However, switching between the internal and external memories is achieved via a switch on the board, so the switching cannot occur while program is running. Figure 3 shows a memory map of KUE-CHIP2.

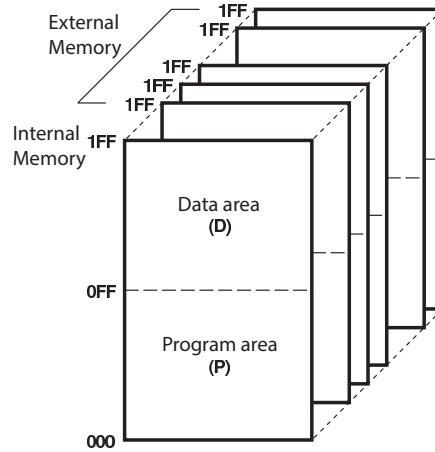


Figure 3: Memory map

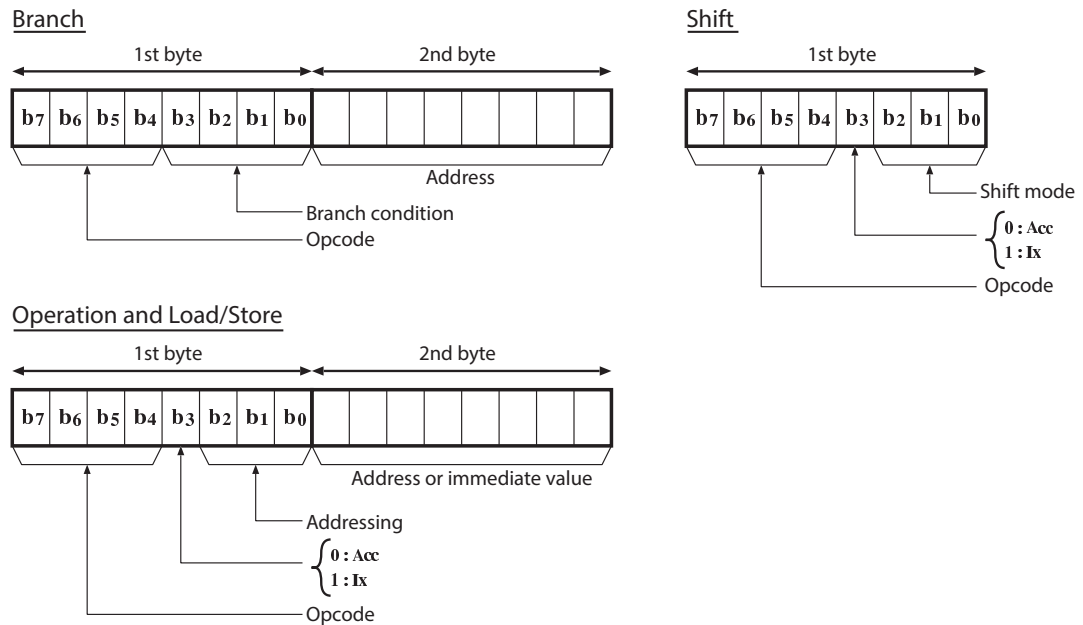


Figure 4: Machine language format

2.3 Instruction Format and Instruction Set

The instructions for KUE-CHIP2 are composed of one byte or two bytes. The upper four or five bits (from eighth to fifth or fourth bit) of the first byte of an instruction specify the type of the instruction.

In branching instructions, the lower four bits represent the branching condition. In shift instructions, the lower three bits show the shift mode. In arithmetic instructions and load/store instructions, the fourth bit specifies the arithmetic register, and the lower three bits (third to first bits) represent the address mode of the second operand. The machine language format of these instructions is shown in Fig. 4. The second byte of the two-byte instruction contains addresses or data. In data, negative numbers are shown in two's complement representation, the most significant bit (eighth bit) corresponding to the sign.

Appendix A provides the instruction set for KUE-CHIP2. There are nineteen types of instructions, including two input and output instructions, two shift instructions, two flag-setting instructions, two arithmetic and logic instructions, a load instruction, a store instruction, a branching instruction, an NOP instruction, and a halt instruction. The instructions vary in length and comprise one- and two-byte formats. There are five memory-access address specification modes: an immediate-value mode, a direct-address mode (program area P, data area D), and an indexed address mode with the index register (IX) (program area P and data area D). In the immediate-value mode, the second byte is data. In the direct-address mode, the second byte shows the address where data is stored. In the indexed address mode, the value determined by adding the content of the second byte to the value of the index register (IX) is the address of the second operand. The simple operation of each instruction is shown in Table 1.

Table 1: Operations of instructions

| | |
|--------------------------|---|
| NOP(No Operation) | Do nothing |
| HLT(HaLT) | Halt |
| OUT(OUTput) | Output the content of ACC to the output port |
| IN(INput) | Input data to ACC from the input port |
| RCF(Reset CF) | Reset the carry flag CF |
| SCF(Set CF) | Set CF |
| Bcc(Branch cc) | Branching instruction by the condition code cc |
| Ssm(Shift sm) | Shift instruction, where sm specifies the shift mode |
| Rsm(Rotate sm) | Rotation shift instruction, where sm specifies the shift mode |
| LD(Load) | Load instruction which moves data from memory to a register |
| ST(STore) | Store instruction which moves the content of a register to memory |
| SBC(SuBtract with Carry) | Subtraction including CF |
| ADC(ADd with Carry) | Addition including CF |
| SUB(SUBtract) | Subtraction not including CF |
| ADD(ADD) | Addition not including CF |
| EOR(Exclusive OR) | Exclusive OR for each bit |
| OR(OR) | OR for each bit |
| AND(AND) | AND for each bit |
| CMP(CoMPare) | Compare instruction |

2.4 Programming and KUE-CHIP2 Operation

KUE-CHIP2 executes a single instruction in up to five phases, P0 to P4. It has an ordinary operation mode as well as a single-instruction mode that halts operation at each instruction, and a single-phase mode that halts operation at each phase. We can use these modes to understand how the circuit is operating at each phase. The changes in register values due to the execution of each phase are carried out after that phase has finished and before going into the next phase.

Table 2 shows the execution phases for each instruction of KUE-CHIP2. All instructions commonly have the same fetching of the instruction in phase P0 and transfer the instruction to the instruction register in phase P1.

Table 2: Instruction execution phases

| phase instruction | | P0 | P1 | P2 | P3 | P4 |
|---|----------------------|---------------------|------------|--|--|--|
| HLT | | (PC) → MAR PC ++ | (Mem) → IR | HALT | | |
| NOP | | | | NOP | | |
| OUT | | | | (ACC) → OBUF | 0 → OBUF_WE | |
| IN | | | | (IBUF) → ACC 0 → IBUF_RE | 0 → IBUF_FLG_CLR | |
| RCF | | | | 0 → CF | | |
| SCF | | | | 1 → CF | | |
| Bcc | | | | (PC) → MAR PC ++ | STATUS CHECK (Mem) → PC (condition satisfied) | |
| Ssm Rsm | | | | TCF SET SHIFT | NF, ZF, VF, CF SET | |
| LD | ACC IX | | | (A) → B | | |
| | d | | | (PC) → MAR PC ++ | (Mem) → A | |
| | [d] (d) | | | | (Mem) → MAR | (Mem) → A |
| | [IX + d] (IX + d) | | | | (IX) → ALU → MAR (Mem) → ALU → MAR | |
| ST | [d] (d) | | | (PC) → MAR PC ++ | (Mem) → MAR | A → (Mem) |
| | [IX + d] (IX + d) | | | | (IX) → ALU → MAR (Mem) → ALU → MAR | |
| SBC ADC SUB ADD EOR OR AND CMP | ACC IX | | | (A) → ALU → A (B) → ALU → A [(CF)] → ALU → A NF, ZF, VF, CF SET | | |
| | d | | | (PC) → MAR PC ++ | (A) → ALU → A (B) → ALU → A [(CF)] → ALU → A NF, ZF, VF, CF SET | |
| | [d] (d) | | | | (Mem) → MAR | (A) → ALU → A (B) → ALU → A [(CF)] → ALU → A NF, ZF, VF, CF SET |
| | [IX + d] (IX + d) | | | | (IX) → ALU → MAR (Mem) → ALU → MAR | |

Phase P0: The instruction is fetched. Specifically, the address where the instruction is entered (stored in the program counter PC) is sent to the memory address register, and then 1 is automatically added to the program counter.

Phase P1: The data whose address is contained in the memory address register is transmitted to the instruction register via the bus DBi.

Phase P2: From this phase onward, the operation differs depending on the instruction. For example, for two-byte instructions, the address of the second byte is transmitted to the memory-address register and a Read or Write operation is performed. Then the program counter PC is incremented by 1.

Here, the actual execution of a simple two-number addition program is considered in clock-phase units. Consider a program that stores the sum of two signed single-precision integers (8-bit data expressed in two's complement representation) stored in D1 (address 80h) and D2 (address 81h) in ANS (address 82h) (the assembler grammar follows Appendix A). The values of D1 and D2 have been preset to

$$\begin{aligned} D1 &= 3 \\ D2 &= -3. \end{aligned}$$

Moreover, the program counter PC is set so that the program executes from address 0h.

List 1

| Address | Data | Label | Instruction | Operand | |
|---------|-------|-------|-------------|-----------|--------------------------|
| | | D1 : | EQU | 80h | Define the address of D1 |
| | | D2 : | EQU | 81h | |
| | | ANS: | EQU | 82h | |
| 00 : | 64 80 | | LD | ACC,[D1] | |
| 02 : | B4 81 | | ADD | ACC,[D2] | |
| 04 : | 74 82 | | ST | ACC,[ANS] | |
| 06 : | 08 | | HLT | | |
| | | | END | | |
| 80 : | 03 | | | | |
| 81 : | FD | | | | |

Tracing the load instruction “LD ACC, [D1]” in each phase just after program execution ($PC = 0$) yields the following (See Table 2, Figs. 5 and 6).

P0 (1) The PC content “0” is transmitted to MAR to read out the instruction indicated by PC at address 0.

The content of MAR then becomes “0.” Next, PC is incremented by 1 so that “PC = 1.”

P1 (2) The instruction content is read out.

The content (64h) at address 00h, specified by the content of MAR, is transmitted to the instruction register IR via the input bus DBi.

P2 (3) The instruction decoder IDC decodes the content of IR.

The upper four bits “0110” are decoded as a load instruction, the “0” at the fourth bit indicates the subject register as ACC, and the “100” in the lower three bits indicates that the address of the data to be loaded is given in the direct-address format, and that the data is in the program area (Fig. 7).

(4) Next, to generate the address of the second operand, the second byte is read out. To do this, the PC content “PC = 1” is again sent to MAR, and PC is incremented by 1 so that “PC = 2.”

The content of MAR becomes “1.”

P3 The memory is read out.

(5) The content at address 01h “80h”, indicated by the content of MAR, is read out, and transferred to MAR via DBi ALU DBo. The content of MAR becomes “80h.”

P4 The second operand is read out.

(6) The program area is indicated for the instruction address mode, so the address for the data to be read out is the most significant bit “0,” which is the address “0 1000 0000” (address 80h). The read-out data (content of address 80h) is stored in ACC via DBi ALU DBo.

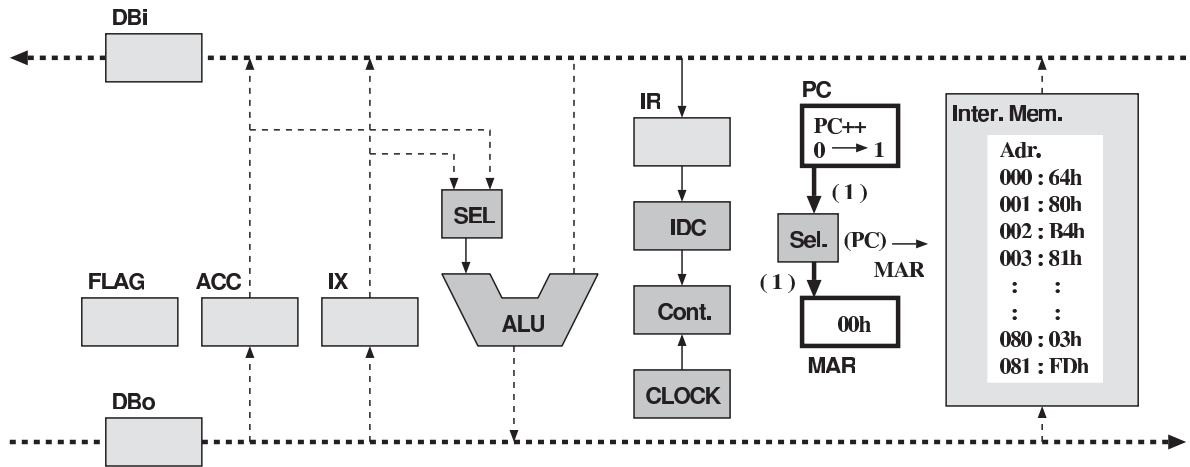
2.5 Program Execution

Power must be supplied to the KUE-CHIP2 board before inputting the program. When supplying the power, sufficient care should be taken to check the output voltage and polarity of the power-supply device. Be careful not to break the board.

2.5.1 Switches and Display Devices on the Board

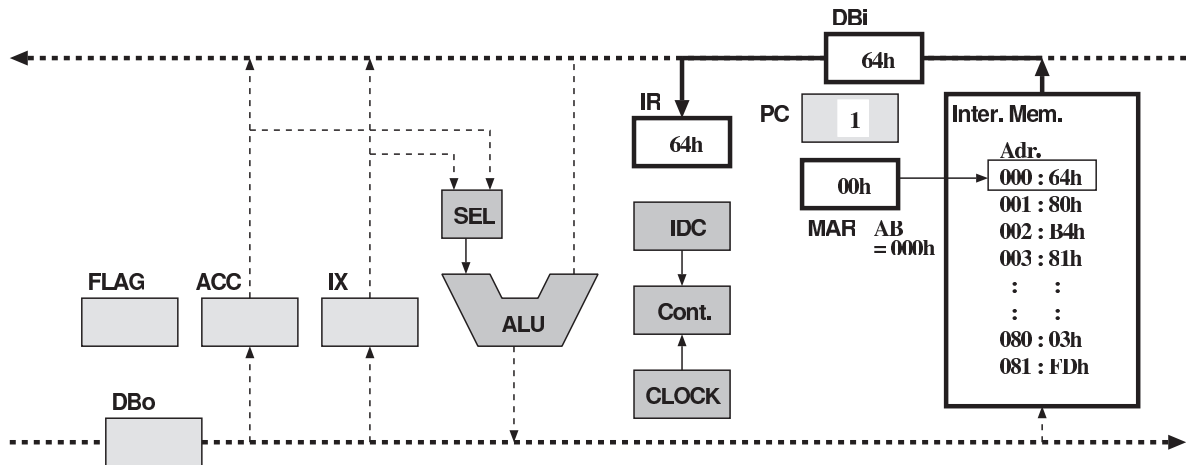
Figure 8 shows an external view of the KUE-CHIP2 board. An overview of the functions of the switches and display devices on the board is as follows.

phase P0 : (PC) \rightarrow MAR , PC++



phase P1 : (Mem) \rightarrow IR

(2) (Mem) \rightarrow IR



phase P2 : (PC) \rightarrow MAR , PC++

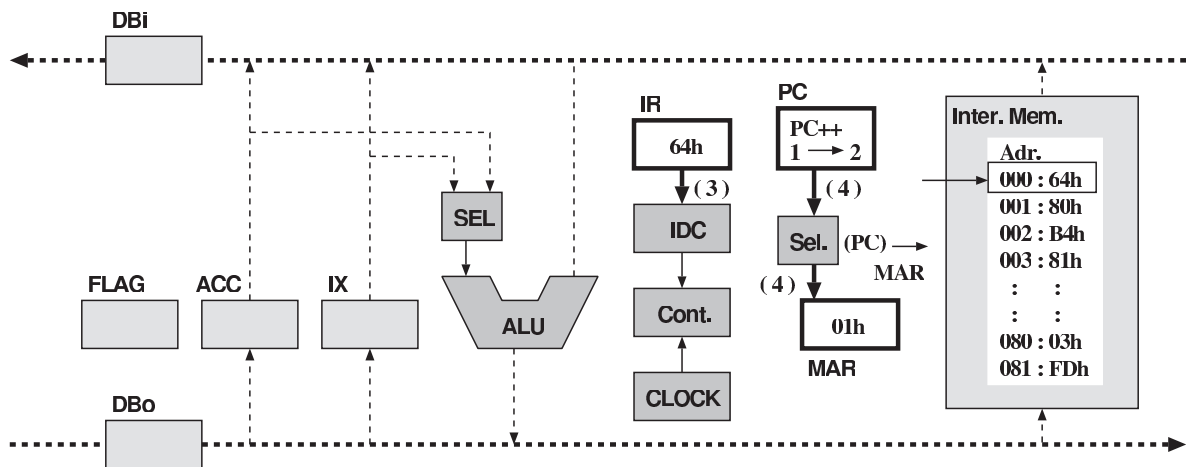


Figure 5: Execution of the Load instruction (1)

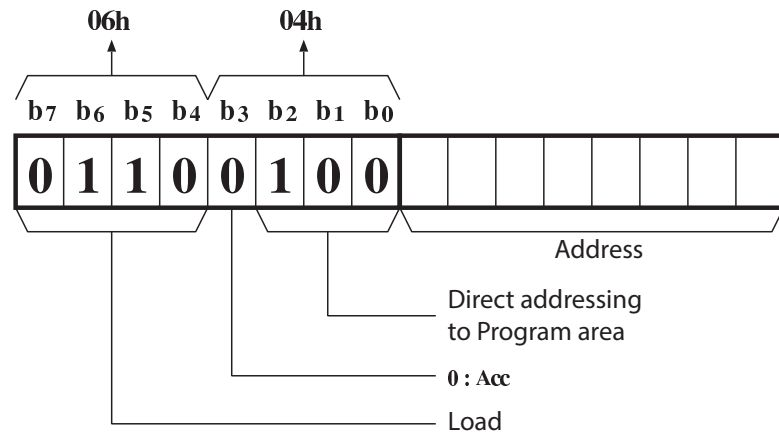


Figure 7: Decoding of Load instruction

(1) Display devices

OP indicates a program or instruction is being executed.

P0–P4 indicates the clock phase currently being executed.

DATA indicates the value of the observation bus. Content is selected with SEL.

ADDRESS indicates the value of the address bus.

IBUF specifies the status of the input buffer.

OBUF specifies the status of the output buffer.

(2) Switches

POWER specifies the power.

RESET resets the entire board.

SEL selects the memory or register for observation and in which data will be written.

DATA sets the data to be written.

ADRINC increments the memory address register by 1.

ADRDEC decreases the memory address register by 1.

MEM switches between the internal memory (INT) and external memory (EXT).

SS executes the program, starting from the address shown in PC until the HLT instruction is given.

If pressed during program execution, the switch halts the program after the completion of the instruction currently being executed.

SI executes one single instruction from the address indicated by PC.

SP executes one phase from the clock phase currently being executed.

CLKFRQ sets the clock frequency for KUE-CHIP2 operation.

(3) Other switch setting

IMC NORMAL

CLK BOARD

WR DATA

IBUF DIPSW

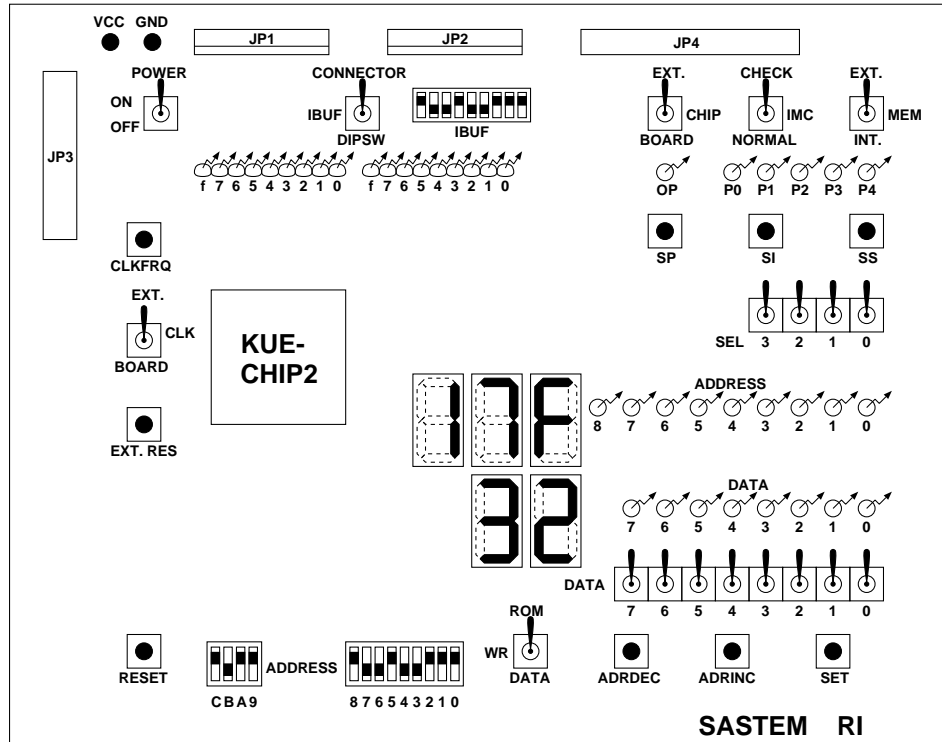


Figure 8: Exterior view of the KUE-CHIP2 board

2.5.2 Example of Input Program

As an easy example, we shall establish whether a program input has been performed correctly by explaining the input, execution, and observation method of a program. The program turns LEDs on one at a time from right to left, to showing the output buffer status. The flow chart in Fig. 9 shows the program overview.

Assuming the program to be input from address 00h, the machine code is List 2. “-” may be either 1 or 0. Here, if 0 is entered, the actual code entered is List 3.

List 2

| Address | Data | | | | Instruction | Operand |
|---------|------|---------|------|------|-------------|----------|
| 00: | 0110 | 001 - | 0000 | 0001 | LD | ACC, 01h |
| 02: | 0001 | 0 - - - | | | OUT | |
| 03: | 0100 | 0111 | | | RLL | ACC |
| 04: | 0011 | 0000 | 0000 | 0010 | BA | 02h |

Table 3: SEL switch selection

| 3 | 2 | 1 | 0 | Selection | |
|---|---|---|---|-----------|-------------------------|
| 0 | 0 | 0 | 0 | MCP | Program area memory |
| 0 | 0 | 0 | 1 | MCD | Data area memory |
| 0 | 0 | 1 | 0 | PC | Program counter |
| 0 | 0 | 1 | 1 | FLAG | FLAG register |
| 0 | 1 | 0 | 0 | ACC | Accumulator |
| 0 | 1 | 0 | 1 | IX | Index register |
| 0 | 1 | 1 | 0 | DBi | Input bus |
| 0 | 1 | 1 | 1 | DBo | Output bus |
| 1 | 0 | 0 | 0 | MAR | Memory address register |
| 1 | 0 | 0 | 1 | IR | Instruction register |

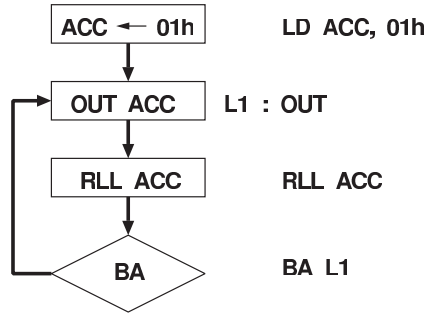


Figure 9: Flow chart of List 2

List 3

| Address | Data | | | | Instruction | Operand |
|---------|------|------|------|------|-------------|----------|
| 00: | 0110 | 0010 | 0000 | 0001 | LD | ACC, 01h |
| 02: | 0001 | 0000 | | | OUT | |
| 03: | 0100 | 0111 | | | RLL | ACC |
| 04: | 0011 | 0000 | 0000 | 0010 | BA | 02h |

2.5.3 Program Input

Next, we introduce how to input the program in List 3.

- (1) Connect the power supply and turn on the power.
Confirm that the power supply output is 5 V and check the polarity of the power supply. Then, first switch on the power supply and then the POWER switch on the KUE-CHIP2 board.
- (2) Set the switches as follows.

| Switch | Position | |
|--------|----------|---|
| CLKFRQ | 8 | : Set the clock to 100 Hz* |
| CLK | BOARD | : Use the internal clock |
| WR | DATA | : Write data from the 8-bit toggle switch |
| CHIP | BOARD | : Use the KUE-CHIP2 on the board |
| IMC | NORMAL | : Assign the lower nine bits of the address bus with MAR of KUE-CHIP2 |
| MEM | INT | : Use the internal memory |

* : If a high-speed clock is used, all eight LEDs will be seen as if they turn on simultaneously rather than sequentially.

- (3) Press the RESET switch.
- (4) Set the SEL switch to “0010” and check the content of PC.
→ All DATA LEDs turn off and the seven-segment LED display shows “00” (PC is “00”).
- (5) Set the SEL switch to “1000” and check the content of MAR.
→ The DATA LED turns off and the seven-segment LED display shows “00.”
- (6) Set the SEL switch to “0000.”
→ The ADDRESS LED does not change, and the DATA LED shows the value (indefinite) of the 00h address in the memory.
- (7) Set the DATA switch to the value “0110 0010,” which is to be written to memory address 00h.

→ The display does not change.

(8) Press the SET switch.

→ The DATA LED displays “0110 0010” (the value has been written into the memory).

(9) Press the ADRINC switch.

→ The ADDRESS LED displays “01h” (MAR has been incremented).

(10) Steps (7) to (9) are repeated until all of the program of List 3 has been input.

2.5.4 Program Execution

Expected operation

Press the RESET switch and then the SS switch.

→ The program operates continuously, so the LEDs indicating the DBo status turn on one by one from right to left.

If the program did not operate

Check the memory.

- (1) If the program is in mid-operation (the yellow OP LED is on, and the P0–P4 LEDs turn on sequentially), press the SS switch to halt the program.
- (2) Press the RESET switch. The RESET switch resets the registers ACC, PC, MAR, etc., but does not reset the content of the memory. Use the SEL switch to check that “PC = 00h” and “MAR = 00h.”
- (3) Using the ADRINC switch and ADRDEC switch, check the content of each address.
- (4) If an error is found, rewrite the content of that address.
- (5) Repeat steps (3) and (4) until the program ends (address 05h).

Trace the program.

- (1) Press the SS switch and halt program execution.
- (2) Press the RESET switch to reset PC and MAR.
- (3) Press the SI switch and execute one single instruction.

→ If the execution is normal, the seven-segment LED display will repeat (00h → 01h →) 02h → 03h → 05h.

- (4) Check the content of ACC.

3 Experiment Tasks

3.1 Trace an Addition Program

Execute the addition program of List 1 in clock-phase units, and trace the observable registers such as FLAG, ACC, IR, and PC at the end of each clock phase. Here, use (1) to (6) for the calculation formula; for (1), trace all observable items with the observation bus OB. Then from (2) to (6), trace the FLAG during the execution of the ADD instruction. For (2) to (6), also trace the FLAG when the ADD instruction is replaced with the ADC instruction. To confirm that the value has been input correctly, execute the program once before tracing to check that the correct addition result is obtained. Make a note of the addition result.

- | | | | |
|-----|---------------|-----|-------------------|
| (1) | $2 + 3 = ?$ | (4) | $2 + (-3) = ?$ |
| (2) | $126 + 1 = ?$ | (5) | $-127 + (-1) = ?$ |
| (3) | $126 + 2 = ?$ | (6) | $-127 + (-2) = ?$ |

3.2 Create a Multiplication Program

Create a program for calculating the product of two unsigned integers of two-byte precision. Check the operation of the program on the KUE-CHIP2 board. The format for recording two-byte-precision integers in the memory is shown in Fig. 10(a). For example, assume the multiplicand is stored at locations 80h and 81h, and the multiplier is stored at 82h and 83h. Also, assume that the operation result is stored at 84h and 85h.

If you have time, create a program for multiplying two signed numbers of two-byte precision, expressed in two's complement representation. For example, "FED4h," the two's complement representation of "-300," is stored as shown in Fig. 10(b).

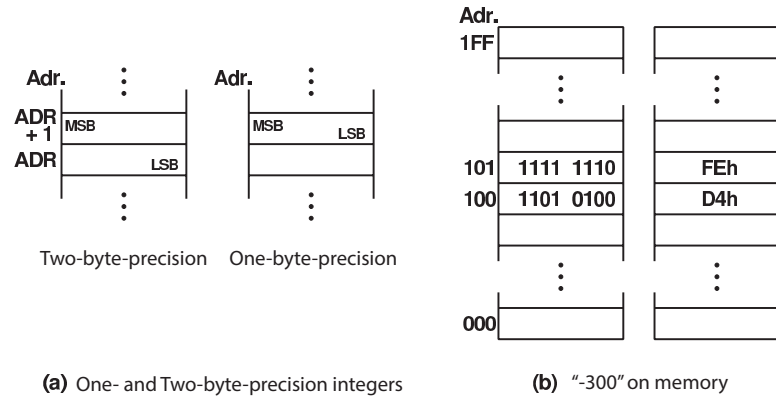


Figure 10: Two-byte precision integer

3.3 Output a Melody (See Appendix B)

- (1) Use the KUE-CHIP2 board to output a square wave at 440 Hz (the musical note "A") to a speaker. The frequency error should be correct to within $\pm 1\%$.
 - (a) Check the frequency of the KUE-CHIP2 board clock using an oscilloscope. The KUE-CHIP2 board clock can be varied using a switch. Use the switch CLKFRQ to check all the clocks from No. 8 to No. 0. To measure the clock frequency, set the CLK switch to the middle position and obtain the clock signal from connector JP3 (pin No. 3).

- (b) Create a program using the frequencies measured in (a) to output a 440 Hz square wave. When making reference to the program in Appendix B, think about the optimal T_0 , a , and b (that is, those that minimize the error).
 - (c) Run the program to check that a 440 Hz ($\pm 1\%$) signal is emitted from the speaker. Also, observe the output waveform using an oscilloscope.
- (2) Create a program that outputs a melody using the KEU-CHIP2 board. The program must be able to handle rests and continuous sounds at the same pitch (simply using the program of List 5 as it is will not achieve this). Then, play a simple melody from the speaker. There is no need to output an entire tune; a part of a tune is sufficient. Prepare the musical score yourself.

4 Discussions

- (1) For the result in “3.1 Trace an addition program,” describe the changes in the values of each register for every phase during execution. Refer to section 2.4 and use “Fig. 1 KUE-CHIP2 Architecture” to explain it simply in writing.
- (2) For “3.2 Create a multiplication program,” compare your algorithm with other students’ programs and summarize it from the perspectives of code size and execution speed. Refer to the program you created and to Table 2 to find and compare the theoretical value of the calculation time estimated from the number of steps (clock) until the completion, with the actual calculation time. In the experiment report, show not only the result, but also the calculation process.
- (3) For “3.3 Output a melody,” consider the following:
 - (a) How did you check that the precision was within $\pm 1\%$? Can you think of another way to check it?
 - (b) What kind of measures do you think would improve the precision even further? Consider measures that would improve the program by using the KUE-CHIP2 board by itself, and measures that would involve connecting other devices to the KUE-CHIP2 board.
 - (c) Discuss the data-representation characteristics for your melody. What other kinds of data representation can you think of? Think about this from a perspective of a data representation that is easy for humans to understand, or that best conveys differences between sounds.
 - (d) Consider whether or not the method for outputting your melody can be used with other processors (for example Z80 or 80286). Justify your answer by investigating the instruction phases of these other processors. If the method cannot be used with the other processors, discuss what kind of issues need to be considered with the other processors to output the melody (square wave).
- (4) If you have time, summarize the architecture of a CPU that you often use (or a well-known CPU), discussing the characteristics of its registers, instruction sets, and memory space. Summarize how the arithmetic instructions are executed.

References

- [1] 神原弘之 他 (H. Kanbara, *et al.*), “KUE-CHIP2 教育用ボードリファレンスマニュアル Version 1.10 (KEU-CHIP2 Educational Board Reference Manual Version 1.10),” 京都高度技術研究所 (Advanced Science, Technology & Management Research Institute of Kyoto), 京都 (Kyoto), 1993.

- [2] 神原弘之 他 (H. Kanbara, *et al.*), “KUE-CHIP2設計ドキュメント Version1.10 (KUE-CHIP2 Design Document Version 1.10),” 京都高度技術研究所 (Advanced Science, Technology & Management Research Institute of Kyoto), 京都 (Kyoto), 1993.
- [3] 国立天文台 編 (National Astronomical Observatory of Japan (Ed.)), “理科年表 (Chronological Scientific Tables),” 丸善 (Maruzen), 東京 (Tokyo), 1993.
- [4] 神原弘之 (H. Kanbara), 安浦寛人 (H. Yasuura), “計算機教育用マイクロコンピュータの開発とその応用 – 集積回路技術を利用した情報工学実験 (Development and Application of an Micro-computer for Computer Education–Information Engineering Experiments Using Integrated Circuit Technology),” 情報処理 (Jouhou Shori), Vol. 33, No. 2, pp. 118–127, 1992.
- [5] 成田福雄 (F. Narita), “Z80 演算サブルーチン・ライブラリ (Z80 Arithmetic Subroutine Library),” pp. 17–40, pp. 61–80, 工学社 (Kohgakusha), 東京 (Tokyo) 1986.
- [6] 西沢昭 (S. Nishizawa), “Z80 上級プログラミング (Z80 Advanced Programming),” pp. 32–69, pp. 176–218, CQ 出版 (CQ Publishing), 東京 (Tokyo), 1984.
- [7] 稲葉保 (T. Inaba), “精選アナログ実用回路集 (Practical Analog Circuit Selection),” pp. 206–215, CQ 出版 (CQ Publishing), 東京 (Tokyo), 1989.
- [8] 横山直隆 (N. Yokoyama), “パソコン・インターフェースの製作自習 (Teach Yourself to Create Computer Interfaces),” pp. 242–265, 技術評論社 (Gijutsu Hyoron Sha), 東京 (Tokyo), 1986.

A KUE-CHIP2 Architecture

Table 4: Address modes

| | |
|----------|--------------------------------|
| ACC | Accumulator |
| IX | Index register |
| d | Immediate-value |
| (d) | Direct address (data area) |
| [d] | Direct address (program area) |
| (IX + d) | Indexed address (data area) |
| [IX + d] | Indexed address (program area) |

Table 5: Logical address modes

| | |
|-------|-------------------------------|
| {ea} | All address modes |
| {reg} | Register address mode |
| {imm} | Immediate data mode |
| {ma} | Memory reference address mode |

Table 6: Address mode correspondence table

| | ACC | IX | d | (d) | [d] | (IX+d) | [IX+d] |
|-------|-----|----|---|-----|-----|--------|--------|
| {ea} | ○ | ○ | ○ | ○ | ○ | ○ | ○ |
| {reg} | ○ | ○ | × | × | × | × | × |
| {imm} | × | × | ○ | × | × | × | × |
| {ma} | × | × | × | ○ | ○ | ○ | ○ |

Table 7: Instruction table

| Abbrev. Code | Address mode | function |
|--------------|--------------|-----------------------------|
| HLT | | Halt |
| NOP | | No OPeration |
| IN | | INput |
| OUT | | OUTput |
| SCF | | Set Carry Flag |
| RCF | | Reset Carry Flag |
| LD | {reg},{ea} | LoaD |
| ST | {reg},{ma} | STore |
| ADD | {reg},{ea} | ADD |
| ADC | {reg},{ea} | ADd with Carry |
| SUB | {reg},{ea} | SUBtract |
| SBC | {reg},{ea} | SuBtract with Carry |
| CMP | {reg},{ea} | CoMPare |
| AND | {reg},{ea} | AND |
| OR | {reg},{ea} | OR |
| EOR | {reg},{ea} | Exclusive OR |
| <i>Ssm</i> | {reg} | Shift |
| <i>Rsm</i> | {reg} | Rotate |
| RA | | Right Arithmetically |
| LA | | Left Arithmetically |
| RL | | Right Logically |
| LL | | Left Logically |
| Bcc | {imm} | Branch |
| A | | Always |
| VF | | on oVerFlow |
| NZ | | on Not Zero |
| Z | | on Zero |
| ZP | | on Zero or Positive |
| N | | on Negative |
| P | | on Positive |
| ZN | | on Zero or Negative |
| NI | | on No Input |
| NO | | on No Output |
| NC | | on No Carry |
| C | | on Carry |
| GE | | on Greater than or Equal to |
| LT | | on Less Than |
| GT | | on Greater Than |
| LE | | on Less than or Equal to |

Table 8: Instruction code

| Code | Instruction code (First byte, second byte) | | | | | | | | Instruction function overview | |
|------|--|---|---|---|---|---|---|---|-------------------------------|----------------|
| NOP | 0 | 0 | 0 | 0 | 0 | — | — | — | × | No OPeration |
| HLT | 0 | 0 | 0 | 0 | 1 | — | — | — | × | Halt |
| | 0 | 1 | 0 | 1 | — | — | — | — | × | Unused (HLT) |
| OUT | 0 | 0 | 0 | 1 | 0 | — | — | — | × | OUTput |
| IN | 0 | 0 | 0 | 1 | 1 | — | — | — | × | INput |
| RCF | 0 | 0 | 1 | 0 | 0 | — | — | — | × | Reset CF |
| SCF | 0 | 0 | 1 | 0 | 1 | — | — | — | × | Set CF |
| Bcc | 0 | 0 | 1 | 1 | | c | c | | ⊙ | Branch cc |
| Ssm | 0 | 1 | 0 | 0 | A | 0 | s | m | × | Shift sm |
| Rsm | 0 | 1 | 0 | 0 | A | 1 | s | m | × | Rotate sm |
| LD | 0 | 1 | 1 | 0 | A | | | B | ○ | LoaD |
| ST | 0 | 1 | 1 | 1 | A | | | B | ⊙ | STore |
| SBC | 1 | 0 | 0 | 0 | A | | | B | ○ | SuB with Carry |
| ADC | 1 | 0 | 0 | 1 | A | | | B | ○ | ADd with Carry |
| SUB | 1 | 0 | 1 | 0 | A | | | B | ○ | SUBtract |
| ADD | 1 | 0 | 1 | 1 | A | | | B | ○ | ADD |
| EOR | 1 | 1 | 0 | 0 | A | | | B | ○ | Exclusive OR |
| OR | 1 | 1 | 0 | 1 | A | | | B | ○ | OR |
| AND | 1 | 1 | 1 | 0 | A | | | B | ○ | AND |
| CMP | 1 | 1 | 1 | 1 | A | | | B | ○ | CoMPare |

Table 9: Condition code (cc)

| | | | | | | |
|----|---|---|---|---|-----------------------------|---|
| A | 0 | 0 | 0 | 0 | Always | Always satisfied |
| VF | 1 | 0 | 0 | 0 | on oVerFlow | Digit overflow $VF = 1$ |
| NZ | 0 | 0 | 0 | 1 | on Not Zero | $\neq 0$ $ZF = 0$ |
| Z | 1 | 0 | 0 | 1 | on Zero | $= 0$ $ZF = 1$ |
| ZP | 0 | 0 | 1 | 0 | on Zero or Positive | ≥ 0 $NF = 0$ |
| N | 1 | 0 | 1 | 0 | on Negative | < 0 $NF = 1$ |
| P | 0 | 0 | 1 | 1 | on Positive | > 0 $(NF \vee ZF) = 0$ |
| ZN | 1 | 0 | 1 | 1 | on Zero or Negative | ≤ 0 $(NF \vee ZF) = 1$ |
| NI | 0 | 1 | 0 | 0 | on No Input | $IBUF_FLG_IN = 0$ |
| NO | 1 | 1 | 0 | 0 | on No Output | $OBUF_FLG_IN = 0$ |
| NC | 0 | 1 | 0 | 1 | on Not Carry | $CF = 0$ |
| C | 1 | 1 | 0 | 1 | on Carry | $CF = 1$ |
| GE | 0 | 1 | 1 | 0 | on Greater than or Equal to | ≥ 0 $(VF \oplus NF) = 0$ |
| LT | 1 | 1 | 1 | 0 | on Less than | < 0 $(VF \oplus NF) = 1$ |
| GT | 0 | 1 | 1 | 1 | on Greater than | > 0 $((VF \oplus NF) \vee ZF) = 0$ |
| LE | 1 | 1 | 1 | 1 | on Less than or Equal to | ≤ 0 $((VF \oplus NF) \vee ZF) = 1$ |

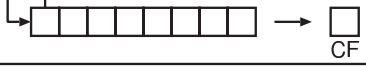
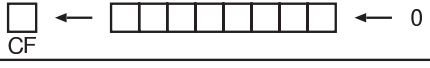

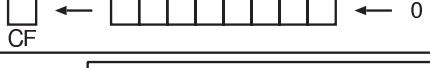
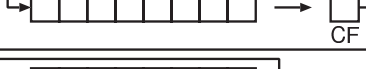
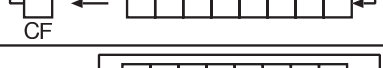
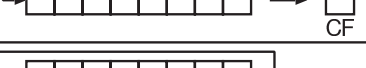
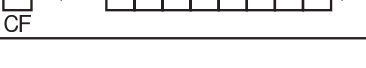
Table 10: Shift mode (sm)

| | | | |
|----|---|---|----------------------|
| RA | 0 | 0 | Right Arithmetically |
| LA | 0 | 1 | Left Arithmetically |
| RL | 1 | 0 | Right Logically |
| LL | 1 | 1 | Left Logically |

Table 11: Instruction code (A, B)

| | |
|---------------------------------------|--|
| $A = 0$: ACC | $B = 000$: ACC |
| $A = 1$: IX | $B = 001$: IX |
| B' (second byte) | $B = 01-$: Immediate value (B' : Data) |
| \times : Unnecessary | $B = 100$: Direct (P) (B' : Address) |
| \bigcirc : Unnecessary or necessary | $B = 101$: Direct (D) (B' : Address) |
| \odot : Necessary | $B = 110$: Indexed (P) ($B' + (IX)$: Address) |
| | $B = 111$: Indexed (D) ($B' + (IX)$: Address) |

Table 12: Shift/Rotate instruction

| Abbrev. | MSB | LSB |
|---------|--|-----|
| SRA |  | CF |
| SLA |  | CF |
| SRL |  | CF |
| SLL |  | CF |
| RRA |  | CF |
| RLA |  | CF |
| RRL |  | CF |
| RLL |  | CF |

B Melody Output

B.1 Sound Output

Sound is emitted by the speaker after outputting a square wave to it from the KUE-CHIP2 board. The speaker circuit configuration is shown in Fig. 11. Here, the speaker is connected to a D/A converter, with 00h and FFh outputting alternately from the KUE-CHIP2 board.

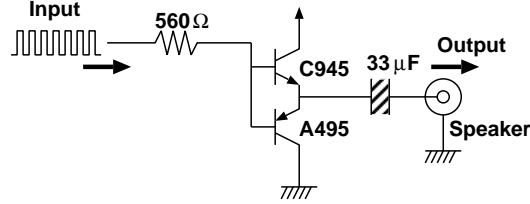


Figure 11: Speaker circuit configuration

List 4 is a program for outputting a square wave. This program outputs a square wave to all the bits of the connector JP1. According to the program, the JP1 connector output is “on” for the addresses 003 to 00B, and “off” for address 000, from addresses 001 to 002, and from addresses 00C to 012. The time T_a required for execution from addresses 002 to 00A at this time is

$$T_a = (12 + 8a) \cdot T_0,$$

where T_0 is the clock frequency. The time T_b required to execute from addresses 000 to 001, and from addresses 00B to 012 is

$$T_b = (16 + 8b) \cdot T_0.$$

The period T of the square wave output to the speaker (Fig. 12) is defined as the sum of T_a and T_b :

$$T = T_a + T_b$$

To adjust the square-wave frequency, the values of a and b can be changed, and meaningless instruction strings such as NOP can be inserted into the loop.

If the values correspond to waves that are within the audible range ($20 \leq 1/T \leq 20,000$ Hz), then it will be produced as a “sound” when output from the speaker. Incidentally, the program in Lists 4 and 5 cannot produce frequencies higher or lower than the audible range, so adjusting the frequency alone cannot produce a silent state.

List 4

| Address | Data | Label | Instruction | Operand | # Phase |
|---------|--------|-------|-------------|----------|---------|
| 000 : | 62 FF | L0 | LD | ACC, FFh | 4 |
| 002 : | 10 | | OUT | | 4 |
| 003 : | 62 a | | LD | ACC, a | 4 |
| 005 : | A2 01 | L1 | SUB | ACC, 01h | 4 |
| 007 : | 31 05 | | BNZ | L1 | 4 |
| 009 : | 62 00 | | LD | ACC, 00h | 4 |
| 00B : | 10 | | OUT | | 4 |
| 00C : | 62 b | | LD | ACC, b | 4 |
| 00E : | A2 01 | L2 | SUB | ACC, 01h | 4 |
| 010 : | 31 0E | | BNZ | L2 | 4 |
| 012 : | 30 00 | | BA | L0 | 4 |

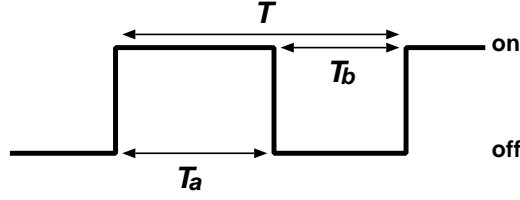


Figure 12: Square-wave output

B.2 Example of a Program to Output a Melody

The program in List 5 outputs a melody consisting of the notes C D E F G A B C for a duration of one second each (the data area is omitted).

The data for each sound comprises three bytes, and corresponds to n_1 , n_2 , and n_3 in the program. The output frequency f and the sound duration T_{beep} at this time are

$$f = \frac{1}{2 \cdot (8 \cdot n_1 + 41) \cdot T_0}$$

$$T_{\text{beep}} = (((8 \cdot n_1 + 41) \cdot n_2 + 28) \cdot n_3 + 39) \cdot T_0.$$

The sound pitch is adjusted via the parameter n_1 , and the sound duration by n_2 and n_3 . Parameter n_3 was introduced because only very short sounds can be produced with n_2 alone. Parameters dptr1 and dptr2 represent the upper and lower limits of the data area. The parameter “image” is the bit image (00h or FFh) that is output. To output square waves, in the program, after the bit image has been referenced, the inverted value is stored. Moreover, dptr is the address of the data that is currently being referenced, and n2 and n3 are the addresses where the above-mentioned respective values n_2 and n_3 are stored. Each of these prepares a storage area within the data area (use the addresses in the data area, and change the underlined addresses to fit your melody).

The frequency for each note in the scale is given in Table 4. The note frequency is doubled in going from one octave to the next. When making changes to the program in List 5, the above parameter calculation formula must also be changed.

Table 13: Frequencies of notes in the scale [Hz]

| | | | |
|----|--------|----|---------------|
| C | 261.63 | G | 392.00 |
| C# | 277.18 | G# | 415.30 |
| D | 293.66 | A | 440.00 |
| D# | 311.13 | A# | 466.16 |
| E | 329.63 | B | 493.88 |
| F | 349.23 | C | 523.25 |
| F# | 369.99 | | |

List 5

| | | | | |
|-------|--------------|-------|-----|--------------|
| 000 : | 62 00 | | LD | ACC, dptr1 |
| 002 : | 75 <u>1A</u> | | ST | ACC, (dptr) |
| 004 : | 65 <u>1A</u> | L0 : | LD | ACC, (dptr) |
| 006 : | 68 | | LD | IX, ACC |
| 007 : | B2 03 | | ADD | ACC, 03h |
| 009 : | 75 <u>1A</u> | | ST | ACC, (dptr) |
| 00B : | A2 <u>18</u> | | SUB | ACC, dptr2 |
| 00D : | 31 13 | | BNZ | L1 |
| 00F : | 62 00 | | LD | ACC, dptr1 |
| 011 : | 75 <u>1A</u> | | ST | ACC, (dptr) |
| 013 : | 67 02 | L1 : | LD | ACC, (IX+2) |
| 015 : | 75 <u>1C</u> | | ST | ACC, (n3) |
| 017 : | 67 01 | Ln3 : | LD | ACC, (IX+1) |
| 019 : | 75 <u>1B</u> | | ST | ACC, (n2) |
| 01B : | 65 <u>19</u> | Ln2 : | LD | ACC, (image) |
| 01D : | 10 | | OUT | |
| 01E : | C2 FF | | EOR | ACC, FFh |
| 020 : | 75 <u>19</u> | | ST | ACC, (image) |
| 022 : | 67 00 | | LD | ACC, (IX+0) |
| 024 : | A2 01 | Ln1 : | SUB | ACC, 01h |
| 026 : | 31 24 | | BNZ | Ln1 |
| 028 : | 65 <u>1B</u> | | LD | ACC, (n2) |
| 02A : | A2 01 | | SUB | ACC, 01h |
| 02C : | 75 <u>1B</u> | | ST | ACC, (n2) |
| 02E : | 31 1B | | BNZ | Ln2 |
| 030 : | 65 <u>1C</u> | | LD | ACC, (n3) |
| 032 : | A2 01 | | SUB | ACC, 01h |
| 034 : | 75 <u>1C</u> | | ST | ACC, (n3) |
| 036 : | 31 17 | | BNZ | Ln3 |
| 038 : | 30 04 | | BA | L0 |