コンピュータ アーキテクチャ I
練習問題

塩谷 亮太 (shioya@ci.i.u-tokyo.ac.jp)

東京大学大学院情報理工学系研究科 創造情報学専攻

練習問題 1．2進数

練習問題 1

- 以下では 2進数は 0b... 16進数は 0x... と表記するものとする
- 以下の１６進数を２進数で表記せよ
  - (1) 0x1111
  - (2) 0xabcd1234
- 以下の２進数を１６進数で表記せよ
  - (3) 0b101011
  - (4) 0b111111111111
- 以下の演算を行え
  - (5) 0b11111110 / 0b10 (「/」 は割り算)

練習問題１解答

- 以下の１６進数を２進数で表記せよ
  - (1) 0x1111 → 0b0001 0001 0001 0001
  - (2) 0xabcd1234 → 0b1010 1011 1100 1101 0001 0010 0011 0100
- 以下の２進数を１６進数で表記せよ
  - (3) 0b10 1011 → 0x2b
  - (4) 0b1111 1111 1111 → 0xfff
- 以下の演算を行え
  - (5) 0b11111110 / 0b10 → 0b1111111

練習問題 2. 命令セット

練習問題２で使用する命令セット

- レジスタ：
  - A, B, C, D, E, F の６つがあるものとする
- 命令：
  - 次のページで説明する li, add, sub, ld, st, b の ６命令がある

練習問題２で使用する命令セット

- li：即値をレジスタに読み込む
  - 例：`li 0 → A　　// レジスタ A に 0 をいれる`
- add：加算
  - 例：`add A,1 → A　// レジスタ A に 1 を足して A に書く`
- sub：減算
  - 例：`sub B,C → D　// レジスタ B から C を引いて D に書く`

練習問題２で使用する命令セット

- ld：ロード命令
  - 例：`ld (B) → A  // レジスタ B の値をアドレスとして`
    `// メモリから値を読み出し A に書き込む`
- st：ストア命令
  - 例：`st A → (B)  // レジスタ B の値をアドレスとして`
    `// そこにレジスタ A の値を書き込む`
- b：条件分岐命令
  - 例：`b A < B, LABEL  // もし A<B であれば LABEL に飛ぶ`
  - 例：`b LABEL  // 条件式がなければ無条件に LABEL に飛ぶ`

練習問題２で使用する命令セット

- ラベルについて
  - C 言語のラベルと同様に，任意の命令の場所を表すためにラベルを使うことができる
  - ラベルには任意の名前をつけることが可能であり，「ラベル名：」で表記する

- 例：
  - ```
      li A←0
    LABEL_EXAMPLE:    // LABEL_EXAMPLE というラベルをここに定義
      add A←A+A
      b LABEL_EXAMPLE // LABEL_EXAMPLE に無条件で飛ぶ
                      // 具体的には add に飛ぶ
    ```
    ```

練習問題 2 (1,2)

- (1) 以下の C 言語で書かれたプログラムをアセンブリ言語で表せ
  ただし，ループに突入時の i<=6 のチェックは省いて良い

- (2) アセンブリ言語で書いたプログラムを実行した際の，実行される
  命令の系列を列挙せよ

```
1: for (i = 0; i <= 6; i+=2) {
2: }
```

練習問題 2 (1,2) 解答

```
1: for (i = 0; i <= 6; i+=2) {
2: }

// 方針：まず if と goto に書き直す
  i = 0;
LOOP:
  i += 2;
  if (i <= 6)
    goto LOOP

// (1) アセンブリ
  li A→0
LOOP:
  add A+2→A
  b A<=6,LOOP

// (2) 実行される命令の系列
// (li/add/b はそれぞれ 1 つしかなくて区別がつくので
//  表記を省略しています)
li,add,b,add,b,add,b,add,b
```

練習問題 2 (3,4)

- (3) 以下の C 言語で書かれたプログラムをアセンブリ言語で表せ
  ただし変数 i の初期値はメモリアドレス 0x100 に格納されており，
  プログラムが終了した際に i の結果が 0x100 に格納されているもの
  とする.

- (4) アセンブリ言語で書いたプログラムを実行した際の，実行される
  命令の系列を列挙せよ

```
1: i = 5;
2: if (i > 3) {
3:     i = i + 1;
4: }
5: else {
6:     i = i - 1;
7: }
```

練習問題 2 (3,4) 解答

```
1: i = 5;
2: if (i > 3) {
3:     i = i + 1;
4: }
5: else {
6:     i = i - 1;
7: }

// 方針：まず if 文を，if と goto に書き直す
  i = 5
  if (i > 3)
    goto LABEL_ADD
  i = i – 1
  goto LABEL_EXIT
ADD:
  i = i + 1;
LABEL_EXIT:
```

練習問題 2 (3,4) 解答

```
// アセンブリ
  li 0x100→A   // アドレス 0x100 から i を読み出す
  ld (A)→B
  b B>3,LABEL_ADD
  sub B-1→B
  b LABEL_EXIT
LABEL_ADD:
  add B+1→B
LABEL_EXIT:
  st B→(A)     // アドレス 0x100 に書き戻し

// 系列（区別がつくところは表記を省略しています）
li, ld, b, add B+1→B, st B→(A)
```

練習問題 3. RISC-V

RISC-V の 基本整数命令

- 概要
  - 加減算，論理演算，
    ロード・ストア，
    即値，分岐とジャンプなど
  - 各命令は 32bit 幅
- 命令のオペランドは
  rd←rs1,rs2 と表記するもの
  とする

RV32I Base Instruction Set

imm[31:12]				rd	0110111	LUI
imm[31:12]				rd	0010111	AUIPC
imm[20|10:1|11|19:12]				rd	1101111	JAL
imm[11:0]		rs1	000	rd	1100111	JALR
imm[12|10:5]	rs2	rs1	000	imm[4:1|11]	1100011	BEQ
imm[12|10:5]	rs2	rs1	001	imm[4:1|11]	1100011	BNE
imm[12|10:5]	rs2	rs1	100	imm[4:1|11]	1100011	BLT
imm[12|10:5]	rs2	rs1	101	imm[4:1|11]	1100011	BGE
imm[12|10:5]	rs2	rs1	110	imm[4:1|11]	1100011	BLTU
imm[12|10:5]	rs2	rs1	111	imm[4:1|11]	1100011	BGEU
imm[11:0]		rs1	000	rd	0000011	LB
imm[11:0]		rs1	001	rd	0000011	LH
imm[11:0]		rs1	010	rd	0000011	LW
imm[11:0]		rs1	100	rd	0000011	LBU
imm[11:0]		rs1	101	rd	0000011	LHU
imm[11:5]	rs2	rs1	000	imm[4:0]	0100011	SB
imm[11:5]	rs2	rs1	001	imm[4:0]	0100011	SH
imm[11:5]	rs2	rs1	010	imm[4:0]	0100011	SW
imm[11:0]		rs1	000	rd	0010011	ADDI
imm[11:0]		rs1	010	rd	0010011	SLTI
imm[11:0]		rs1	011	rd	0010011	SLTIU
imm[11:0]		rs1	100	rd	0010011	XORI
imm[11:0]		rs1	110	rd	0010011	ORI
imm[11:0]		rs1	111	rd	0010011	ANDI
0000000	shamt	rs1	001	rd	0010011	SLLI
0000000	shamt	rs1	101	rd	0010011	SRLI
0100000	shamt	rs1	101	rd	0010011	SRAI
0000000	rs2	rs1	000	rd	0110011	ADD
0100000	rs2	rs1	000	rd	0110011	SUB
0000000	rs2	rs1	001	rd	0110011	SLL
0000000	rs2	rs1	010	rd	0110011	SLT
0000000	rs2	rs1	011	rd	0110011	SLTU
0000000	rs2	rs1	100	rd	0110011	XOR
0000000	rs2	rs1	101	rd	0110011	SRL
0100000	rs2	rs1	101	rd	0110011	SRA
0000000	rs2	rs1	110	rd	0110011	OR
0000000	rs2	rs1	111	rd	0110011	AND
0000 | pred	succ	00000	000	00000	0001111	FENCE
0000 | 0000	0000	00000	001	00000	0001111	FENCE.I
000000000000		00000	000	00000	1110011	ECALL
000000000001		00000	000	00000	1110011	EBREAK
csr		rs1	001	rd	1110011	CSRRW
csr		rs1	010	rd	1110011	CSRRS
csr		rs1	011	rd	1110011	CSRRC
csr		zimm	101	rd	1110011	CSRRWI
csr		zimm	110	rd	1110011	CSRRSI
csr		zimm	111	rd	1110011	CSRRCI

画像は下記より
The RISC-V Instruction Set Manual Volume I: User-Level ISA Document Version 2.2

練習問題３

- RISC-V の「and x1←x2,x3」命令を２進数で表記すると以下の通りとなる
  0000000 00011 00010 111 00001 0110011

  - (1) 上記を or x1←x2,x3 に書き換え，２進数で表記せよ

  - (2) 上記を add x2←x3,x4 に書き換え，２進数で表記せよ

練習問題３ 解答

- RISC-V の「and x1←x2,x3」命令を２進数で表記すると以下の通りとなる
  0000000 00011 00010 111 00001 0110011

  - (1) 上記を or x1←x2,x3 に書き換え，２進数で表記せよ
    and: 0000000 00011 00010 111 00001 0110011
    or: 0000000 00011 00010 110 00001 0110011

  - (2) 上記を add x2←x3,x4 に書き換え，２進数で表記せよ
    and: 0000000 00011 00010 111 00001 0110011
    or: 0000000 00100 00011 111 00010 0110011

練習問題 4 論理ゲート

練習問題４

- 良い練習問題の案が浮かばないため，保留

練習問題 5 パイプライン

練習問題５

- パイプライン・プロセッサを構成することを考える
  - in-order スカラ・プロセッサであるものとする
  - 各命令は以下の５つのステージを経て実行されるものとする
    - F：フェッチ，D：デコード，X：演算，
      M：メモリアクセス，W：書き戻し
    - これらの各処理には 1 nano second (ns) がかかるものとする
    - 演算子しか行わずメモリアクセスを伴わない命令でも，必ず M を経るものとする
  - 処理に必要な時間とは，最初の命令の F の開始から，最後の命令の W が終了するまでの時間である

練習問題５

- (1) 以下の命令列を実行するのに必要な時間をパイプラインチャートを描いて求めよ
  - ただしレジスタアクセスにおけるフォワーディングを行うものとする
  - それ以外の必要な場合のみパイプラインを適宜ストールして実行するものとする

```
ld   x2←(x1)
ld   x3←(x1)
add x4←x6+x7
add x5←x2+x3
```

- (2) 上記においてフォワーディングを行わなかった場合に必要な時間を求めよ

練習問題５

- (3) 以下の命令列を最後実行するのに必要な時間を計算せよ
  - プロセッサは分岐予測を行うものとし，b 命令が分岐予測ミスを起こして次の命令を実行したあと，フラッシュされてやり直した場合を想定せよ
  - ただしレジスタアクセスにおけるフォワーディングを行うものとする
  - それ以外の必要な場合のみパイプラインを適宜ストールして実行するものとする

```
  li x1←1
  ld x2←(x3)
  b x1==x2, LABEL
  add x1←x2+x3
LABEL:
  add x2←x3+x4
```

練習問題５ (1) 解答

- (1) 以下の命令列を実行するのに必要な時間をパイプラインチャートを描いて求めよ
  - ただしレジスタアクセスにおけるフォワーディングを行うものとする
  - それ以外の必要な場合のみパイプラインを適宜ストールして実行するものとする

```
// 下記のように実行されるため，8ns かかる
ld     x2←(x1)    F D X M W
ld     x3←(x1)      F D X M W
add x4←x6+x7          F D X M W
add x5←x2+x3            F D X M W
```

- (2) 上記においてフォワーディングを行わなかった場合に必要な時間をパイプラインチャートを描いて求めよ
  - // 下記のように実行されるため，10ns かかる
    // 2命令目の W が終わるまで4命令目の D を待つためストール

```
ld     x2←(x1)    F D X M W
ld     x3←(x1)      F D X M W
add x4←x6+x7          F D X M W
add x5←x2+x3            F _ _ D X M W
```

練習問題５ (2) 解答

- (3) 以下の命令列を最後実行するのに必要な時間を計算せよ
  - プロセッサは分岐予測を行うものとし，b 命令が分岐予測ミスを起こして次の命令を実行したあと，フラッシュされてやり直した場合を想定せよ
  - ただしレジスタアクセスにおけるフォワーディングを行うものとする
  - それ以外の必要な場合のみパイプラインを適宜ストールして実行するものとする

```
// 13ns
li x1←1           F D X M W
ld x2←(x3)          F D X M W
b x1==x2, LABEL       F D _ X M W
add x1←x2+x3            F _ D X M W          // フラッシュされる
add x2←x3+x4              F D X M W          //
add x2←x3+x4                    F D X M W    // この命令はやりなおし
```

練習問題 6 性能モデル

ハザードと実行時間

- 以下のようにおいた場合，
  - 理想的な実行の際のサイクル数：Ct
  - 何らかのハザードの発生回数：　$Nh = Ni \times Pi \times Ph$
    - 実行命令数：　Ni
    - ハザードを起こす命令の出現率：　Pi
    - その命令毎のハザード発生率：　Ph
  - ハザード時のサイクル数の増加：Cp
- 実行サイクル数 Cr は，理想サイクル数 Ct に対して，
  - $Cr = Ct + Nh \times Cp$

IPC で考えると

- 最終的な性能を考える上で IPC の方が都合がよい
- 実行サイクル数 Cr を命令数 Ni で正規化すると，
  - $\frac{Cr}{Ni} = \frac{Ct}{Ni} + \frac{(Nh \times Cp)}{Ni} = \frac{Ct}{Ni} + \frac{(Ni \times Pi \times Ph \times Cp)}{Ni} = \frac{Ct}{Ni} + Pi \times Ph \times Cp$
- IPC は命令数を実行サイクル数で割ったもの = つまり上記の逆数
  - $IPCr = \frac{1}{\frac{Ct}{Ni} + Pi \times Ph \times Cp} = \frac{1}{\frac{1}{IPCt} + Pi \times Ph \times Cp}$
  - ここで $IPCr$ は実際の IPC，$IPCt$ は理想 IPC

一般化できる

- ハザード a，ハザード b，ハザード c・・・とあった時
  - それぞれおきた回数が $Nh..$，ペナルティが Cp … とする
- 実行サイクル数 Cr は，理想サイクル数 Ct に対して，
  - $Cr = Ct + Nha \times Cpa + Nhb \times Cpb + Nhc \times Cpc + \cdots$
- 実行命令数を Ni とすると，$IPCr = Ni/Cr$

練習問題 6

- 以下のような条件を考える
  - 10段のパイプラインを持つ 2-way スーパスカラプロセッサであり，理想的には IPC=2 で実行できる
  - 全実行命令におけるロード命令の出現率は 0.2
    - ロード命令のキャッシュミス率は 0.1
    - キャッシュミスが発生すると 10 サイクル実行時間が伸びるものとする
  - 全実行命令における分岐命令の出現率は 0.2
    - 分岐予測ミス率は 0.3
    - 分岐予測ミスが発生するとペナルティにより 9 サイクル実行時間が伸びるものとする
- (1) この CPU の IPC を求めよ
- (2) 分岐予測ミス率を 0.2 に改善した場合の IPC を求めよ

練習問題 7 キャッシュ

練習問題 7 キャッシュ

- アドレスの幅が 16 bit，ラインサイズ8B，４エントリのキャッシュについて考える

- 連想度を以下の様に変えた場合に，
  - 1（ダイレクトマップ）
  - 2
  - 4（フルアソシアティブ）

- 以下のようなアドレスによる 1B のアクセスがあった場合を考える
  1. `0x8010, 0x8010, 0x8020, 0x8030, 0x8000, 0x8010, 0x8020, 0x8030`
  2. `0x0088, 0x0099, 0x00AA, 0x00BB, 0x0088, 0x0099, 0x00AA, 0x00BB`
  3. `0x8000, 0x9011, 0x8022, 0x9033, 0x9044, 0xA055, 0x9066, 0x8077`

練習問題 7

- (1) 上記それぞれの場合で，アクセスが全て終わった後のキャッシュの状態（タグの中身）を示せ
  - ４エントリのタグにそれぞれ何が残っているかを，
    連想度3パターン×アクセス系列3パタン= 9 パターン分答える
- (2) 上記それぞれの場合のヒット率を計算せよ
- (3) 各アクセスにおけるヒット時に，それが空間的局所性と時間的局所性のいずれによるのかを分類して答えよ

課題 １０ ダイレクトマップの場合 の系列１

1. 0x8010, 0x8010, 0x8020, 0x8030, 0x8000, 0x8010, 0x8020, 0x8030
   黒字はタグ，赤字はインデックス，緑字はライン

   1. 0x8010 = 0b1000 0000 0001 0000 miss
   2. 0x8010 = 0b1000 0000 0001 0000 hit 時間的局所性
   3. 0x8020 = 0b1000 0000 0010 0000 miss
   4. 0x8030 = 0b1000 0000 0011 0000 miss
   5. 0x8000 = 0b1000 0000 0000 0000 miss
   6. 0x8010 = 0b1000 0000 0001 0000 miss
   7. 0x8020 = 0b1000 0000 0010 0000 miss
   8. 0x8030 = 0b1000 0000 0011 0000 miss
      0b100 0000 0001=0x401

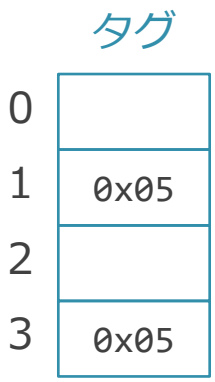
	タグ
0	0x401
1	
2	0x401
3	

- (1) 左の通り
- (2) ヒット率：1/8=0.125
- (3) 上の通り

課題 １０ ダイレクトマップの場合 の系列２

1. 0x0088, 0x0099, 0x00AA, 0x00BB, 0x0088, 0x0099, 0x00AA, 0x00BB

    1. 0x0088 = 0b0000 0000 1000 1000 miss
    2. 0x0099 = 0b0000 0000 1001 1001 miss
    3. 0x00AA = 0b0000 0000 1010 1010 miss
    4. 0x00BB = 0b0000 0000 1011 1011 miss
    5. 0x0088 = 0b0000 0000 1000 1000 miss
    6. 0x0099 = 0b0000 0000 1001 1001 miss
    7. 0x00AA = 0b0000 0000 1010 1010 miss
    8. 0x00BB = 0b0000 0000 1011 1011 miss



- (1) ミス
- (2) ヒット率：0
- (3) 全部ミスなので局所性がない

課題 １０ ダイレクトマップの場合 の系列３

1. 0x8000, 0x9011, 0x8022, 0x9033, 0x9044, 0xA055, 0x9066, 0x8077

    1. 0x8000 = 0b1000 0000 0000 0000 miss
    2. 0x9011 = 0b1001 0000 0001 0001 miss
    3. 0x8022 = 0b1000 0000 0010 0010 miss
    4. 0x9033 = 0b1001 0000 0011 0011 miss
    5. 0x9044 = 0b1001 0000 0100 0100 miss
    6. 0xA055 = 0b1010 0000 0101 0101 miss
    7. 0x9066 = 0b1001 0000 0110 0110 miss 0b100 1000 0011=0x483
    8. 0x8077 = 0b1000 0000 0111 0111 miss 0b100 0000 0011=0x403

タグ

0	0x483
1	
2	0x403
3	

- (1) 左の通り
- (2) ヒット率：1/8=0.125
- (3) 上の通り

課題 １０
2-way セットアソシアティブの場合 の系列１

1. 0x8010, 0x8010, 0x8020, 0x8030, 0x8000, 0x8010, 0x8020, 0x8030
   黒字はタグ，赤字はインデックス，緑字はライン

   1. 0x8010 = 0b1000 0000 0001 0000 miss
   2. 0x8010 = 0b1000 0000 0001 0000 hit 時間的局所性
   3. 0x8020 = 0b1000 0000 0010 0000 miss
   4. 0x8030 = 0b1000 0000 0011 0000 miss
   5. 0x8000 = 0b1000 0000 0000 0000 miss
   6. 0x8010 = 0b1000 0000 0001 0000 miss
   7. 0x8020 = 0b1000 0000 0010 0000 miss
   8. 0x8030 = 0b1000 0000 0011 0000 miss
      0b1000 0000 0011=0x803

タグ

0	0x801	0x802
1		

- (1) 左の通り
- (2) ヒット率：1/8=0.125
- (3) 上の通り

課題 １０
2-way セットアソシアティブの場合 の系列２

1. 0x0088, 0x0099, 0x00AA, 0x00BB, 0x0088, 0x0099, 0x00AA, 0x00BB

    1. 0x0088 = 0b0000 0000 1000 1000 miss
    2. 0x0099 = 0b0000 0000 1001 1001 miss
    3. 0x00AA = 0b0000 0000 1010 1010 miss
    4. 0x00BB = 0b0000 0000 1011 1011 miss
    5. 0x0088 = 0b0000 0000 1000 1000 miss
    6. 0x0099 = 0b0000 0000 1001 1001 miss
    7. 0x00AA = 0b0000 0000 1010 1010 miss
    8. 0x00BB = 0b0000 0000 1011 1011 miss

タグ

0		
1	0x06	0x07

- (1) 左の通り
- (2) ヒット率：0
- (3) 全部ミスなので局所性がない

課題 １０
2-way セットアソシアティブの場合 の系列３

1. 0x8000, 0x9011, 0x8022, 0x9033, 0x9044, 0xA055, 0x9066, 0x8077

    1. 0x8000 = 0b1000 0000 0000 0000 miss
    2. 0x9011 = 0b1001 0000 0001 0001 miss
    3. 0x8022 = 0b1000 0000 0010 0010 miss
    4. 0x9033 = 0b1001 0000 0011 0011 miss
    5. 0x9044 = 0b1001 0000 0100 0100 miss
    6. 0xA055 = 0b1010 0000 0101 0101 miss
    7. 0x9066 = 0b1001 0000 0110 0110 miss 0b1001 0000 0110=0x906
    8. 0x8077 = 0b1000 0000 0111 0111 miss 0b1000 0000 0111=0x807

タグ

0	0x906	0x807
1		

- (1) 左の通り
- (2) ヒット率：0
- (3) 上の通り

課題 １０
フルアソシアティブの場合 の系列１

1. 0x8010, 0x8010, 0x8020, 0x8030, 0x8000, 0x8010, 0x8020, 0x8030
   黒字はタグ，赤字はインデックス，緑字はライン

   1. 0x8010 = 0b1000 0000 0001 0000 miss
   2. 0x8010 = 0b1000 0000 0001 0000 hit 時間的局所性
   3. 0x8020 = 0b1000 0000 0010 0000 miss
   4. 0x8030 = 0b1000 0000 0011 0000 miss
   5. 0x8000 = 0b1000 0000 0000 0000 miss
   6. 0x8010 = 0b1000 0000 0001 0000 hit 時間的局所性
   7. 0x8020 = 0b1000 0000 0010 0000 hit 時間的局所性
   8. 0x8030 = 0b1000 0000 0011 0000 hit 時間的局所性
      0b1 0000 0000 0110=0x1006

タグ

0x1002	0x1004	x01006	0x1000

- (1) 左の通り
- (2) ヒット率：4/8=0.5
- (3) 上の通り

課題 １０
フルアソシアティブの場合 の系列２

1. 0x0088, 0x0099, 0x00AA, 0x00BB, 0x0088, 0x0099, 0x00AA, 0x00BB

1. 0x0088 = 0b0000 0000 1000 1000 miss 0b0 0000 0001 0001=0x11
2. 0x0099 = 0b0000 0000 1001 1001 miss 0b0 0000 0001 0011=0x13
3. 0x00AA = 0b0000 0000 1010 1010 miss 0b0 0000 0001 0101=0x15
4. 0x00BB = 0b0000 0000 1011 1011 miss 0b0 0000 0001 0111=0x17
5. 0x0088 = 0b0000 0000 1000 1000 miss 0b0 0000 0001 0001=0x11 hit 時間
6. 0x0099 = 0b0000 0000 1001 1001 miss 0b0 0000 0001 0011=0x13 hit 時間
7. 0x00AA = 0b0000 0000 1010 1010 miss 0b0 0000 0001 0101=0x15 hit 時間
8. 0x00BB = 0b0000 0000 1011 1011 miss 0b0 0000 0001 0111=0x17 hit 時間

10.

タグ

0x11	0x13	0x15	0x17

- (1) 左のようになる
- (2) ヒット率：4/8=0.5
- (3) 上の通り

課題 １０
フルアソシアティブの場合 の系列３

1. 0x8000, 0x9011, 0x8022, 0x9033, 0x9044, 0xA055, 0x9066, 0x8077

    1. 0x8000 = 0b1000 0000 0000 0000 miss
    2. 0x9011 = 0b1001 0000 0001 0001 miss
    3. 0x8022 = 0b1000 0000 0010 0010 miss
    4. 0x9033 = 0b1001 0000 0011 0011 miss
    5. 0x9044 = 0b1001 0000 0100 0100 miss 0b1 0010 0000 1000=0x1208
    6. 0xA055 = 0b1010 0000 0101 0101 miss 0b1 0100 0000 1010=0x140A
    7. 0x9066 = 0b1001 0000 0110 0110 miss 0b1 0010 0000 1100=0x120A
    8. 0x8077 = 0b1000 0000 0111 0111 miss 0b1 0000 0000 1110=0x100E
    9.

タグ

0x1208	0x140A	0x120A	0x100E

- (1) 左の通り
- (2) ヒット率：0
- (3) 上の通り