
# 論理設計
# 東野担当14回目
# 授業スライド
# １月13日４限

基礎工学部情報科学科　東野輝夫

# 授業計画

- 授業計画（変更）：
  1. ドントケアを含む論理関数の簡単化（６章）
  2. フリップフロップとレジスタ（１０章）
  3. 同期式順序回路（Mealy型, Moore型順序回路）（１１章）
  4. カルノー図を用いた論理関数の簡単化（１章から５章の復習）
  5. 組合せ論理回路設計、よく用いられる組み合わせ回路（７章，８章）
  6. 加減算器とALU、順序回路の簡単化（９章，１２章前半）
  7. 演習
  8. 順序回路の簡単化、カウンタ（１２章後半，１３章）
  9. 中間試験（１章～１１章）
  10. ＩＣを用いた順序回路の実現（１５章）
  11. 演習
  12. ＣＰＵの設計（付録）
  13. ＣＰＵの設計，演習
  14. 乗算器と除算器（１４章）
  15. １月２０日は授業なし（この日は対面での試験の実施日）
  16. 期末試験（１２章～１５章，付録）（期末試験は１月２７日に実施）

期末試験も中間試験同様
オンラインで実施予定

# 質問について

- メールで随時問い合わせや質問にお答えしますので，何かあれば，higashino@ist.osaka-u.ac.jp までメールで質問して下さい.
- また、時間を決めてZoomなどを用いて質問にお答えすることも可能ですので、まずはメールで疑問点や問い合わせ事項などを連絡して下さい。

# お願い

本テキストや授業のビデオなどの電子ファイルを他人に転送したり、ネットへアップロードすることなどを禁止します。

## 著作権保護

- この授業のテキスト（教科書）や授業スライド、授業ビデオの著作権保護に努めて下さい.
- この授業のビデオやスナップショットを録画したり，それらを他の人に転送したり，インターネット上で公開したりすることを禁止します.
- この授業で利用するスライドにはオーム社の教科書の図などが含まれているので，著作権保護の観点から，この授業スライドの公開につながる行為は謹んでください.
- 来年度は CLE を使ったメディア授業でなく，対面の授業ができることを期待していますが，今年度の演習課題の解答が事前に公開されたりすると，来年度の授業で同じ演習課題が使えなくなり，授業テキストの大幅な修正が必要になるため，協力をお願いします.


# 先週のレポート課題解答

# 少し異なる回路のCPU設計

- 付録のCPUと少し異なる下記の図１の回路で構成されるCPUを設計したい．この回路のCPUについて，下記の各問に答えよ．
- 解答はCLEにアップする解答用紙を印刷して記入するか，同様の様式で各自の解答を記入して，CLEにアップしてください．

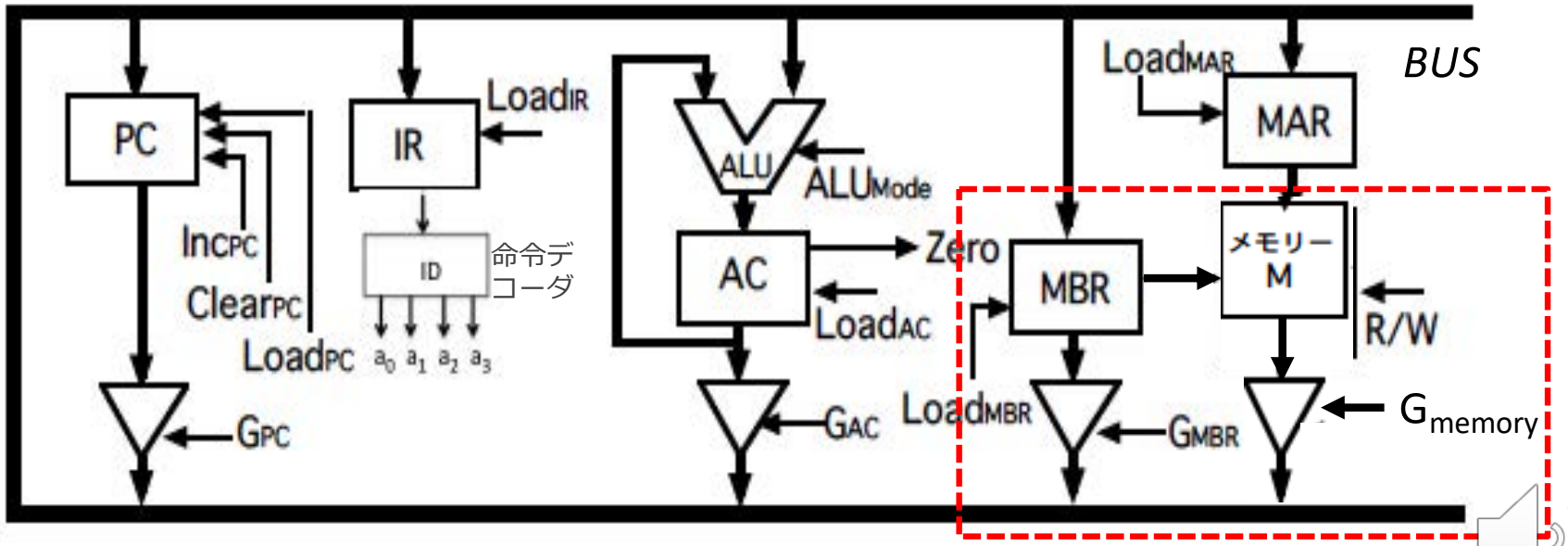
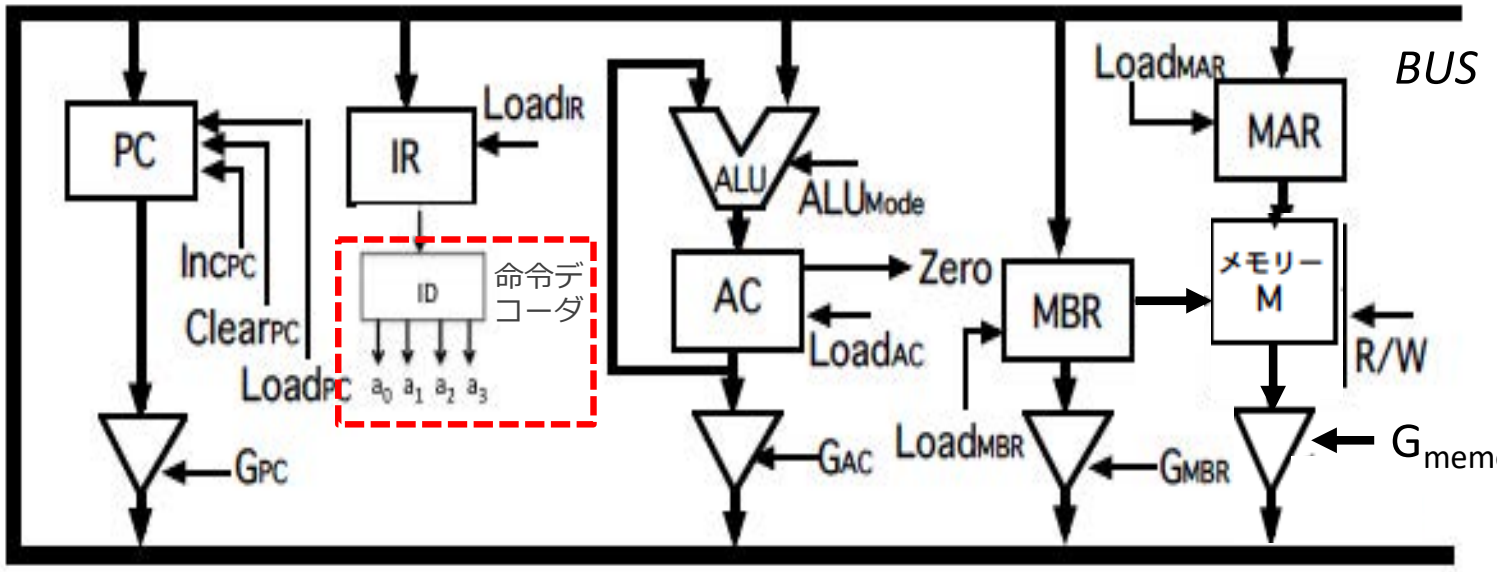



図１　CPUの回路構成

# 少し異なる回路のCPU設計

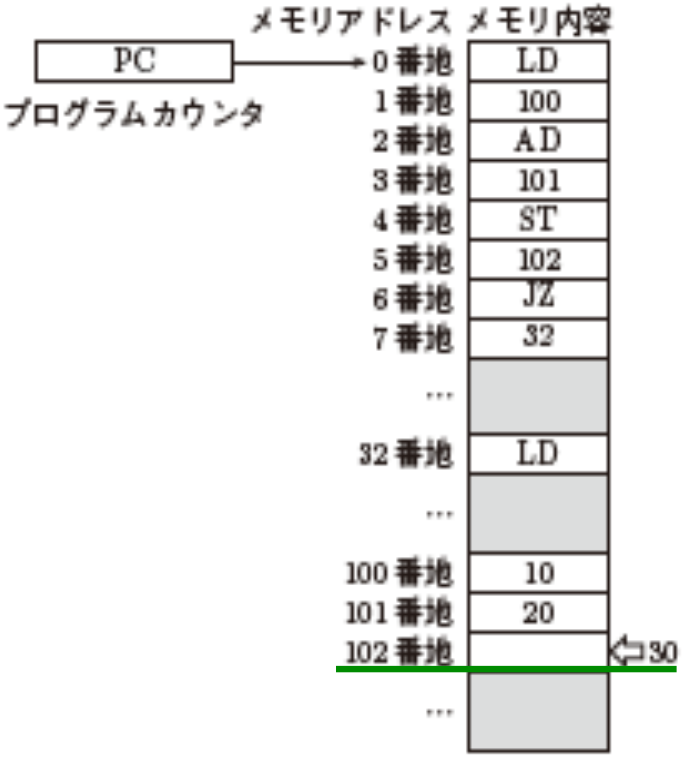
- このCPUの命令は下記の 4 命令のみで，1 語は 8 ビットで，各命令は命令コード，オペランドの順に格納された 2 語とする．レジスタは 8 ビットで，メモリーは 256 バイトとする．また，命令レジスタIRの命令デコーダ ID の$a_0$,$a_1$,$a_2$,$a_3$ 信号の値は下記の通り．

| 命令コード | 記号 | オペランド | 処理 | | IDの出力 |
|---|---|---|---|---|---|
| 00000000 | LD | ADR | AC <- M[ADR] | PC←PC+2 | a0=1, a1=0, a2=0, a3=0 |
| 00000001 | ST | ADR | M[ADR] <- AC | PC←PC+2 | a0=0, a1=1, a2=0, a3=0 |
| 00000010 | AD | ADR | AC <- AC + M[ADR] | PC←PC+2 | a0=0, a1=0, a2=1, a3=0 |
| 00000011 | JZ | B | if AC=0 then PC<-B<br>else PC <-PC + 2 | | a0=0, a1=0, a2=0, a3=1 |



# メモリに格納された機械語プログラム
## （テキスト210頁のアセンブラと同じ）

- 図A·1 にメモリに格納された機械語プログラムの例を示す．この例では，最初プログラムカウンタ PC は 0 番地を指しているものとする．0 番地の命令がロード（LD）命令で 1 番地に 100 が書かれているので，この 2 語命令を実行すると，100 番地の内容 10 がアキュムレータ AC に格納され，プログラムカウンタ PC の値が 2 増加する．
- PC が 2 番地の AD 命令を指し 3 番地に 101 が書かれているので，AD 命令が実行され，アキュムレータ AC に 101 番地の内容 20 が加算される（30 に変化）．
- その後プログラムカウンタ PC の値が 2 増加して 4 になり，5 番地に 102 が書かれているので，4 番地の ST 命令が実行され，アキュムレータ AC の値 30 が 102 番地に格納される．この命令が実行された後，プログラムカウンタ PC の値が 2 増加して 6 になる．
- 6 番地の命令は条件付ジャンプ（JZ）命令なので，AC = 0 かどうかがチェックされる．AC の値は 30 で条件（AC = 0）は偽になるので，7 番地に書かれた 32 番地にはジャンプせず，プログラムカウンタ PC の値はこれまでと同じように 2 だけ増加して 8 になり，8 番地以降の機械語が順次実行されていく．

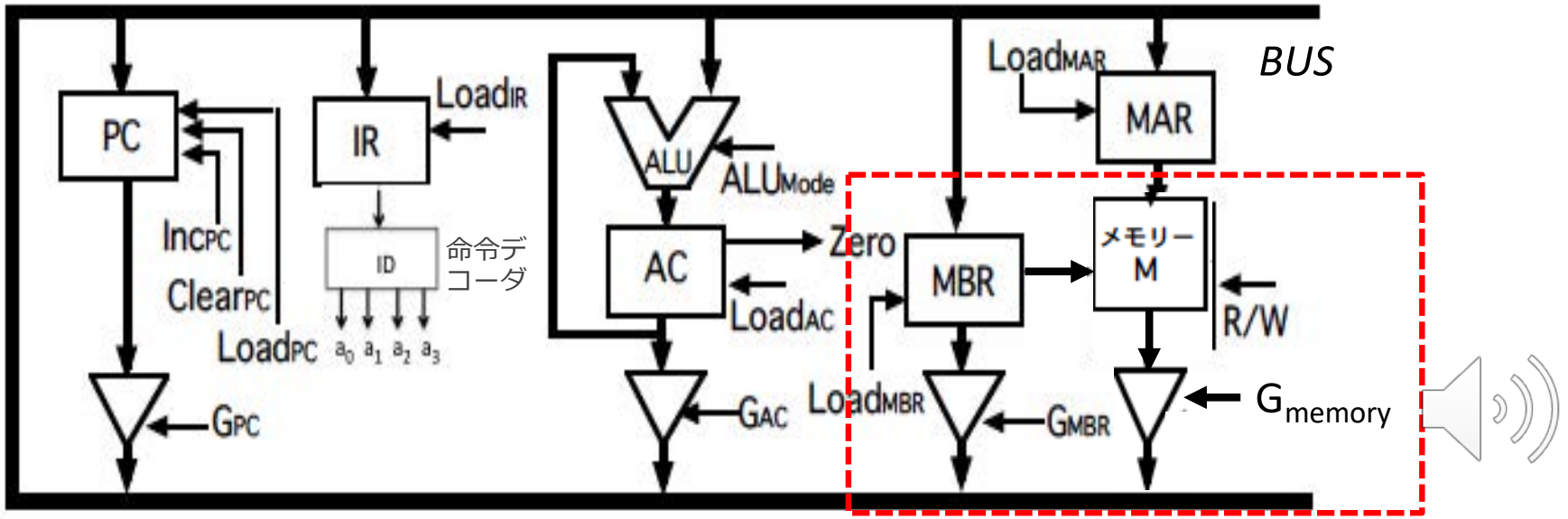



図A・1 メモリに格納された機械語プログラム（アセ…記で表現）

# 少し異なる回路のCPU設計

（問題１）

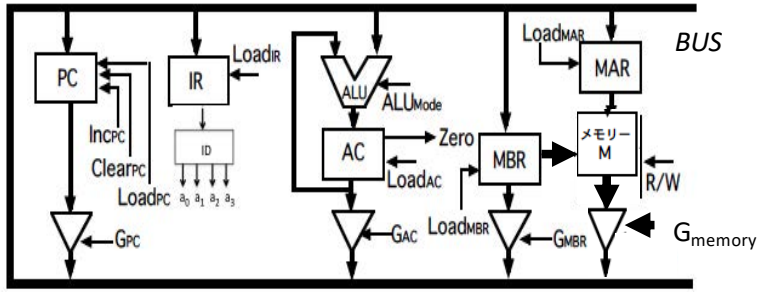
- 下記のLD命令の表記法にならって，ST, AD, JZの各命令の実行サイクルのマイクロ操作列を記載せよ（t0,t1,t2は共通のフェッチサイクル．t3以降の実行サイクルを記載せよ）．
  - フェッチサイクル　t0　MAR ← PC
    t1　IR ← M[MAR]
    t2　PC ← PC+1
  - 実行サイクル　t3　MAR ← PC
    t4　MAR ← M[MAR]
    t5　AC ← M[MAR]
    t6　PC ← PC+1



# 少し異なる回路のCPU設計

（問題１解答）

- 下記のLD命令の表記法にならって，ST, AD, JZの各命令の実行サイクルのマイクロ操作列を記載せよ（t0,t1,t2は共通のフェッチサイクル．t3以降の実行サイクルを記載せよ）．
  - フェッチサイクル　t0　MAR ← PC
    - t1　IR ← M[MAR]
    - t2　PC ← PC+1
  - 実行サイクル　t3　MAR ← PC
    - t4　MAR ← M[MAR]
    - t5　AC ← M[MAR]
    - t6　PC ← PC+1




| | LD | ST | AD | JZ |
|---|---|---|---|---|
| t3 | MAR ← PC | MAR ← PC | MAR ← PC | MAR ← PC |
| t4 | MAR ← M[MAR] | MAR ← M[MAR] | MAR ← M[MAR] | PC ← PC+1 |
| t5 | AC ← M[MAR] | MBR ← AC | AC ← AC+M[MAR] | Test(AC=0)? |
| t6 | PC ← PC+1 | M[MAR] ← MBR | PC ← PC+1 | PC ← M[MAR] |
| t7 | | PC ← PC+1 | | |

# 少し異なる回路のCPU設計

（問題２）

- このCPU回路の制御部をマイクロプログラム方式で実現することを考える．マイクロ命令の形式は次のような17ビットのものとする．

| 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| $Inc_{PC}$ | $Clear_{PC}$ | $Load_{PC}$ | $G_{PC}$ | $Load_{IR}$ | $ALU_{Mode}$ | $Load_{AC}$ | $G_{AC}$ | $Load_{MBR}$ | $G_{MBR}$ | $Load_{MAR}$ | R/W | $G_{Memory}$ | $Inc_{\mu PC}$ | $Clr_{\mu PC}$ | $Load_{\mu PC}$ | Tp |

このマイクロ命令は図2(a)のμP-ROMに格納する．μA-ROMは命令デコーダの値（命令xx）に対し，その命令のマイクロ操作列が書かれたμP-ROMの開始番地Adr_xxが出力される（図2(b)参照）．

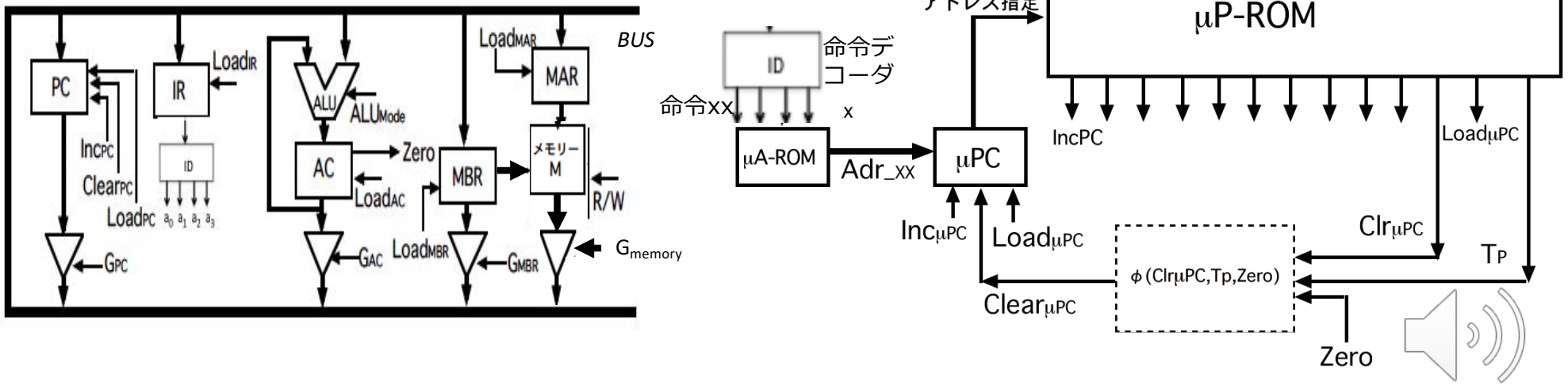



図２(a)：μP-ROMのアドレス指定

# 少し異なる回路のCPU設計

（問題２）

- $Clear_{\mu PC}$信号はμP-ROMの$Clr_{\mu PC}$信号（ビット15）とTp信号（JZ命令の分岐で利用する），ACの値が0 のとき 1 となるZero信号を用いた関数Φ($Clr_{\mu PC}$,Tp,Zero)で指定する．$Clr_{\mu PC}$信号は JZ命令 以外の命令の各実行サイクルの最後のマイクロ操作を実行した際に 1 にすることでμPCの値を 0 に戻す．
- マイクロプログラムカウンタμPCは$Clear_{\mu PC}$=1のとき μPC ← 0 となり，$Load_{\mu PC}$=1 のとき μPC ← Adr_xx となる．$Clear_{\mu PC}$=0 且つ $Inc_{\mu PC}$=1 のとき μPC ← μPC+1 となる．
- 教科書のJZ命令のマイクロ操作列と同様，¬(AC=0) のときに μPC の値を 0 に戻す場合，Tp の値を 1 にして Tp∧¬Zero が 1 になるときに μPC の値が 0 になるように指定する．

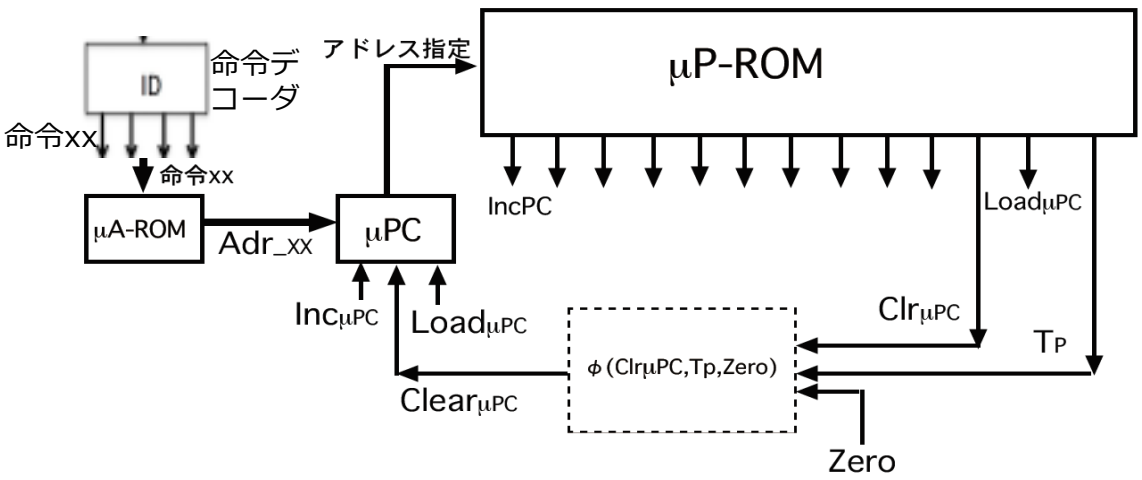



図２(a)：μP-ROMのアドレス指定

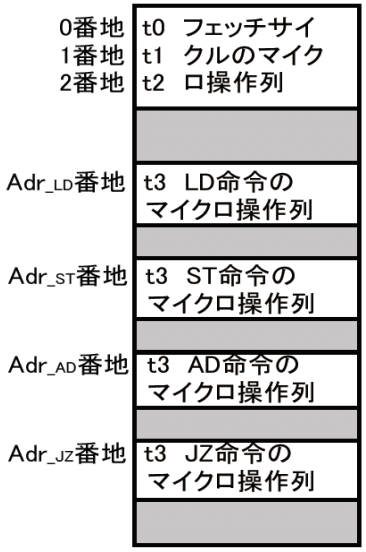



図２(b)：μP-ROMの内容

# 少し異なる回路のCPU設計

（問題２解答）

- この μP-ROM の 0 番地から 2 番地の内容と，ST，AD, JZ 命令のマイクロ操作列（17ビットの命令の列）の内容を下記のような表形式で作成せよ．表中の各セルには，0, 1, ×（ドント・ケア）のいずれかを記入すること．

| | | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| マイクロ操作列 | 番地 | IncPC | ClearPC | LoadPC | GPC | LoadIR | ALUMode | LoadAC | GAC | LoadMBR | GMBR | LoadMAR | R/W | GMemory | IncμPC | ClrμPC | LoadμPC | Tp |
| MAR ← PC | 0番地 | 0 | 0 | 0 | 1 | 0 | × | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 0 | 0 |
| IR ← M[MAR] | 1番地 | 0 | 0 | 0 | 0 | 1 | × | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 0 |
| PC ← PC+1 | 2番地 | 1 | 0 | 0 | 0 | 0 | × | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 |
| | | : | : | : | : | : | : | : | : | : | : | : | : | : | : | : | : | : |
| MAR ← PC | Adr_LD番地 | 0 | 0 | 0 | 1 | 0 | × | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 0 | 0 |
| MAR ← M[MAR] | Adr_LD+1番地 | 0 | 0 | 0 | 0 | 0 | × | 0 | 0 | 0 | 0 | 1 | 0 | 1 | 1 | 0 | 0 | 0 |
| AC ← M[MAR] | Adr_LD+2番地 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 0 |
| PC ← PC+1 | Adr_LD+3番地 | 1 | 0 | 0 | 0 | 0 | × | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 |
| | | : | : | : | : | : | : | : | : | : | : | : | : | : | : | : | : | : |
| MAR ← PC | Adr_ST番地 | 0 | 0 | 0 | 1 | 0 | × | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 0 | 0 |
| MAR ← M[MAR] | Adr_ST+1番地 | 0 | 0 | 0 | 0 | 0 | × | 0 | 0 | 0 | 0 | 1 | 0 | 1 | 1 | 0 | 0 | 0 |
| MBR ← AC | Adr_ST+2番地 | 0 | 0 | 0 | 0 | 0 | × | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 |
| M[MAR] ← MBR | Adr_ST+3番地 | 0 | 0 | 0 | 0 | 0 | × | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 1 | 0 | 0 | 0 |
| PC ← PC+1 | Adr_ST+4番地 | 1 | 0 | 0 | 0 | 0 | × | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 |
| | | : | : | : | : | : | : | : | : | : | : | : | : | : | : | : | : | : |
| MAR ← PC | Adr_AD番地 | 0 | 0 | 0 | 1 | 0 | × | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 0 | 0 |
| MAR ← M[MAR] | Adr_AD+1番地 | 0 | 0 | 0 | 0 | 0 | × | 0 | 0 | 0 | 0 | 1 | 0 | 1 | 1 | 0 | 0 | 0 |
| AC ← AC+M[MAR] | Adr_AD+2番地 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 0 |
| PC ← PC+1 | Adr_AD+3番地 | 1 | 0 | 0 | 0 | 0 | × | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 |
| | | : | : | : | : | : | : | : | : | : | : | : | : | : | : | : | : | : |
| MAR ← PC | Adr_JZ番地 | 0 | 0 | 0 | 1 | 0 | × | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 0 | 0 |
| PC ← PC+1 | Adr_JZ+1番地 | 1 | 0 | 0 | 0 | 0 | × | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 |
| Test(AC=0)? | Adr_JZ+2番地 | 0 | 0 | 0 | 0 | 0 | × | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 1 |
| PC ← M[MAR] | Adr_JZ+3番地 | 0 | 0 | 1 | 0 | 0 | × | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 1 | 0 | 0 |

# 少し異なる回路のCPU設計

（問題２解答）

- この μP-ROM の 0 番地から 2 番地の内容と，ST，AD, JZ 命令のマイクロ操作列（17ビットの命令の列）の内容を下記のような表形式で作成せよ．表中の各セルには，0, 1, ×（ドント・ケア）のいずれかを記入すること．

| | | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| マイクロ操作列 | 番地 | IncPC | ClearPC | LoadPC | GPC | LoadIR | ALUMode | LoadAC | GAC | LoadMBR | GMBR | LoadMAR | R/W | GMemory | IncμPC | ClrμPC | LoadμPC | Tp |
| MAR ← PC | 0番地 | 0 | 0 | 0 | 1 | 0 | × | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 0 | 0 |
| IR ← M[MAR] | 1番地 | 0 | 0 | 0 | 0 | 1 | × | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 0 |
| PC ← PC+1 | 2番地 | 1 | 0 | 0 | 0 | 0 | × | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 |
| | | : | : | : | : | : | : | : | : | : | : | : | : | : | : | : | : | : |
| MAR ← PC | Adr_LD番地 | 0 | 0 | 0 | 1 | 0 | × | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 0 | 0 |
| MAR ← M[MAR] | Adr_LD+1番地 | 0 | 0 | 0 | 0 | 0 | × | 0 | 0 | 0 | 0 | 1 | 0 | 1 | 1 | 0 | 0 | 0 |
| AC ← M[MAR] | Adr_LD+2番地 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 0 |
| PC ← PC+1 | Adr_LD+3番地 | 1 | 0 | 0 | 0 | 0 | × | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 |
| | | : | : | : | : | : | : | : | : | : | : | : | : | : | : | : | : | : |
| MAR ← PC | Adr_ST番地 | 0 | 0 | 0 | 1 | 0 | × | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 0 | 0 |
| MAR ← M[MAR] | Adr_ST+1番地 | 0 | 0 | 0 | 0 | 0 | × | 0 | 0 | 0 | 0 | 1 | 0 | 1 | 1 | 0 | 0 | 0 |
| MBR ← AC | Adr_ST+2番地 | 0 | 0 | 0 | 0 | 0 | × | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 |
| M[MAR] ← MBR | Adr_ST+3番地 | 0 | 0 | 0 | 0 | 0 | × | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 1 | 0 | 0 | 0 |
| PC ← PC+1 | Adr_ST+4番地 | 1 | 0 | 0 | 0 | 0 | × | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 |
| | | : | : | : | : | : | : | : | : | : | : | : | : | : | : | : | : | : |
| MAR ← PC | Adr_AD番地 | 0 | 0 | 0 | 1 | 0 | × | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 0 | 0 |
| MAR ← M[MAR] | Adr_AD+1番地 | 0 | 0 | 0 | 0 | 0 | × | 0 | 0 | 0 | 0 | 1 | 0 | 1 | 1 | 0 | 0 | 0 |
| AC ← AC+M[MAR] | Adr_AD+2番地 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 0 |
| PC ← PC+1 | Adr_AD+3番地 | 1 | 0 | 0 | 0 | 0 | × | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 |
| | | : | : | : | : | : | : | : | : | : | : | : | : | : | : | : | : | : |
| MAR ← PC | Adr_JZ番地 | 0 | 0 | 0 | 1 | 0 | × | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 0 | 0 |
| PC ← PC+1 | Adr_JZ+1番地 | 1 | 0 | 0 | 0 | 0 | × | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 |
| Test(AC=0)? | Adr_JZ+2番地 | 0 | 0 | 0 | 0 | 0 | × | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 1 |
| PC ← M[MAR] | Adr_JZ+3番地 | 0 | 0 | 1 | 0 | 0 | × | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 1 | 0 | 0 |

# 少し異なる回路のCPU設計

（問題２解答）

- この μP-ROM の 0 番地から 2 番地の内容と，ST，AD, JZ 命令のマイクロ操作列（17ビットの命令の列）の内容を下記のような表形式で作成せよ．表中の各セルには，0, 1, ×（ドント・ケア）のいずれかを記入すること．

| | | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| マイクロ操作列 | 番地 | IncPC | ClearPC | LoadPC | GPC | LoadIR | ALUMode | LoadAC | GAC | LoadMBR | GMBR | LoadMAR | R/W | GMemory | IncμPC | ClrμPC | LoadμPC | Tp |
| MAR ← PC | 0番地 | 0 | 0 | 0 | 1 | 0 | × | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 0 | 0 |
| IR ← M[MAR] | 1番地 | 0 | 0 | 0 | 0 | 1 | × | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 0 |
| PC ← PC+1 | 2番地 | 1 | 0 | 0 | 0 | 0 | × | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 |
| | | : | : | : | : | : | : | : | : | : | : | : | : | : | : | : | : | : |
| MAR ← PC | Adr_LD番地 | 0 | 0 | 0 | 1 | 0 | × | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 0 | 0 |
| MAR ← M[MAR] | Adr_LD+1番地 | 0 | 0 | 0 | 0 | 0 | × | 0 | 0 | 0 | 0 | 1 | 0 | 1 | 1 | 0 | 0 | 0 |
| AC ← M[MAR] | Adr_LD+2番地 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 0 |
| PC ← PC+1 | Adr_LD+3番地 | 1 | 0 | 0 | 0 | 0 | × | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 |
| | | : | : | : | : | : | : | : | : | : | : | : | : | : | : | : | : | : |
| MAR ← PC | Adr_ST番地 | 0 | 0 | 0 | 1 | 0 | × | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 0 | 0 |
| MAR ← M[MAR] | Adr_ST+1番地 | 0 | 0 | 0 | 0 | 0 | × | 0 | 0 | 0 | 0 | 1 | 0 | 1 | 1 | 0 | 0 | 0 |
| MBR ← AC | Adr_ST+2番地 | 0 | 0 | 0 | 0 | 0 | × | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 |
| M[MAR] ← MBR | Adr_ST+3番地 | 0 | 0 | 0 | 0 | 0 | × | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 1 | 0 | 0 | 0 |
| PC ← PC+1 | Adr_ST+4番地 | 1 | 0 | 0 | 0 | 0 | × | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 |
| | | : | : | : | : | : | : | : | : | : | : | : | : | : | : | : | : | : |
| MAR ← PC | Adr_AD番地 | 0 | 0 | 0 | 1 | 0 | × | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 0 | 0 |
| MAR ← M[MAR] | Adr_AD+1番地 | 0 | 0 | 0 | 0 | 0 | × | 0 | 0 | 0 | 0 | 1 | 0 | 1 | 1 | 0 | 0 | 0 |
| AC ← AC+M[MAR] | Adr_AD+2番地 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 0 |
| PC ← PC+1 | Adr_AD+3番地 | 1 | 0 | 0 | 0 | 0 | × | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 |
| | | : | : | : | : | : | : | : | : | : | : | : | : | : | : | : | : | : |
| MAR ← PC | Adr_JZ番地 | 0 | 0 | 0 | 1 | 0 | × | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 0 | 0 |
| PC ← PC+1 | Adr_JZ+1番地 | 1 | 0 | 0 | 0 | 0 | × | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 |
| Test(AC=0)? | Adr_JZ+2番地 | 0 | 0 | 0 | 0 | 0 | × | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 1 |
| PC ← M[MAR] | Adr_JZ+3番地 | 0 | 0 | 1 | 0 | 0 | × | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 1 | 0 | 0 |

# 少し異なる回路のCPU設計

（問題３／問題３解答）

- $Clear_{\mu PC}=\Phi(Clr_{\mu PC},Tp,Zero)$ の $\Phi(Clr_{\mu PC},Tp,Zero)$ の論理式を $Clr_{\mu PC},Tp,Zero$ からなる論理関数で表せ.

（解答）

  - $Clear_{\mu PC}=\varphi(Clr_{\mu PC},Tp,Zero)= Clr_{\mu PC} \vee (Tp \wedge \neg Zero)$

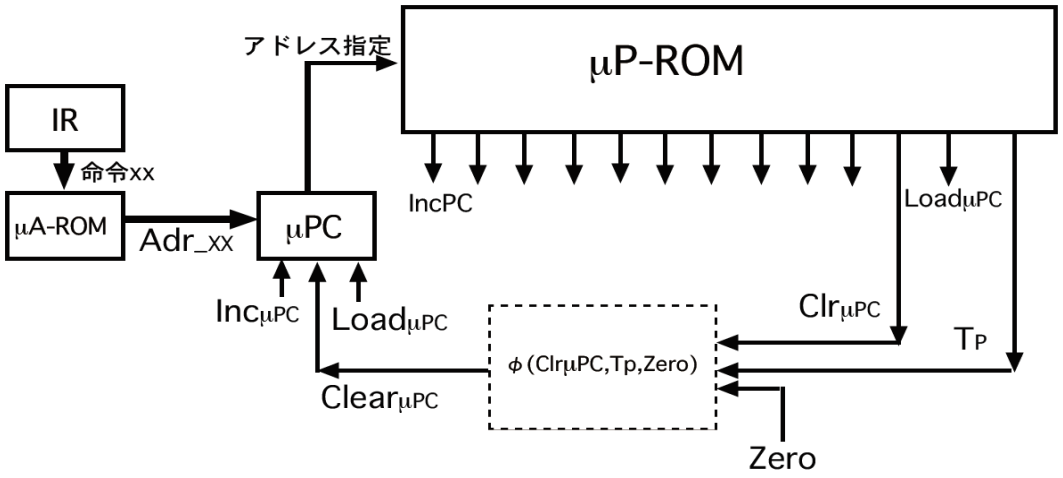


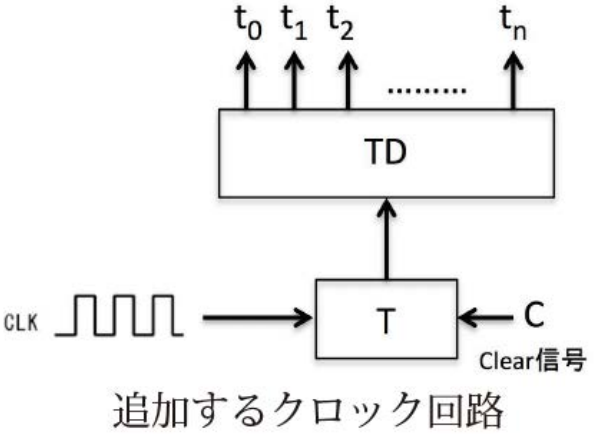

図２(a)：μP-ROMのアドレス指定


図２（b）：μP-ROMの内容

# 少し異なる回路のCPU設計

（問題４）

- 図2(a)の回路に右図のクロック回路（時刻 $t_i$ に TDの $t_i$ 信号が 1 になる）を加えた場合に，信号線 $Inc_{PC}$, $Load_{PC}$ の論理式をそれぞれ $t_0,t_1,\cdots,t_n$, $a_0,a_1,a_2,a_3$,Zeroなどの論理式で表せ.
- なおクロック T の Clear信号 C は $Clear_{\mu PC}$ を用いるものとする.



追加するクロック回路

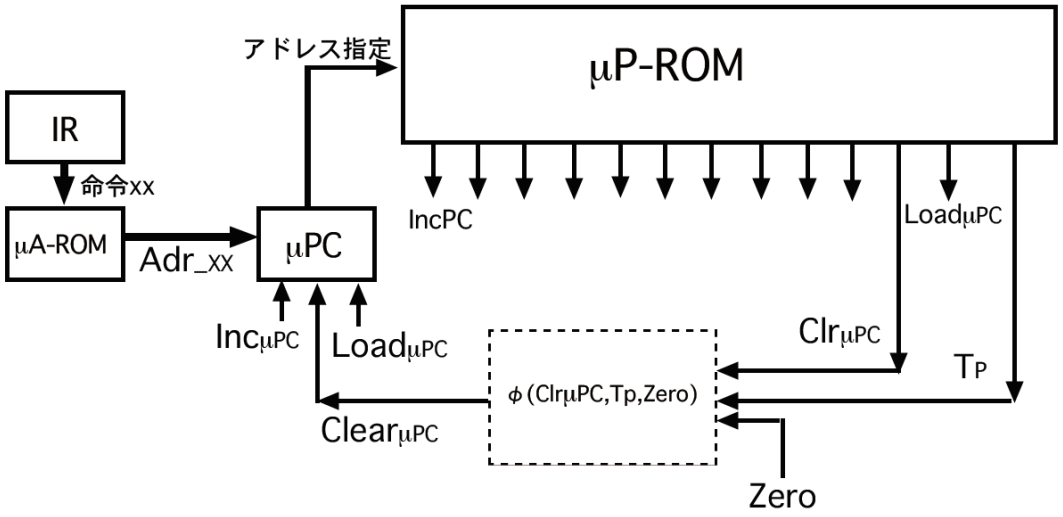


図２(a)：μP-ROMのアドレス指定

図２(b)：μP-ROMの内容

# 少し異なる回路のCPU設計

（問題４解答）

- 図2(a)の回路に右図のクロック回路（時刻 $t_i$ にTDの $t_i$ 信号が 1 になる）を加えた場合に，信号線 $Inc_{PC}$, $Clear_{PC}$ の論理式をそれぞれ $t_0,t_1,\cdots,t_n$, $a_0,a_1,a_2,a_3$,Zeroなどの論理式で表せ.

（解答）

- $Inc_{PC} = t_2 \vee ((a_0 \vee a_2) \wedge t_6) \vee (a_1 \wedge t_7) \vee (a_3 \wedge t_4)$
- $Load_{PC} = a_3 \wedge t_6$

| | | | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | マイクロ操作列 | 番地 | IncPC | ClearPC | LoadPC | GPC | LoadIR | ALUMode | LoadAC | GAC | LoadMBR | GMBR | LoadMAR | R/W | GMemory | IncμPC | ClrμPC | LoadμPC | Tp |
| | MAR ← PC t0 | 0番地 | 0 | 0 | 0 | 1 | 0 | × | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 0 | 0 |
| | IR ← M[MAR] t1 | 1番地 | 0 | 0 | 0 | 0 | 1 | × | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 0 |
| | PC ← PC+1 t2 | 2番地 | 1 | 0 | 0 | 0 | 0 | × | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 |
| | | | : | : | : | : | : | : | : | : | : | : | : | : | : | : | : | : | : |
| $a_0$ | MAR ← PC t3 | Adr_LD番地 | 0 | 0 | 0 | 1 | 0 | × | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 0 | 0 |
| | MAR ← M[MAR] t4 | Adr_LD+1番地 | 0 | 0 | 0 | 0 | 0 | × | 0 | 0 | 0 | 0 | 1 | 0 | 1 | 1 | 0 | 0 | 0 |
| | AC ← M[MAR] t5 | Adr_LD+2番地 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 0 |
| | PC ← PC+1 t6 | Adr_LD+3番地 | 1 | 0 | 0 | 0 | 0 | × | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 |
| | | | : | : | : | : | : | : | : | : | : | : | : | : | : | : | : | : | : |
| $a_1$ | MAR ← PC t3 | Adr_ST番地 | 0 | 0 | 0 | 1 | 0 | × | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 0 | 0 |
| | MAR ← M[MAR] t4 | Adr_ST+1番地 | 0 | 0 | 0 | 0 | 0 | × | 0 | 0 | 0 | 0 | 1 | 0 | 1 | 1 | 0 | 0 | 0 |
| | MBR ← AC t5 | Adr_ST+2番地 | 0 | 0 | 0 | 0 | 0 | × | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 |
| | M[MAR] ← MBR t6 | Adr_ST+3番地 | 0 | 0 | 0 | 0 | 0 | × | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 1 | 0 | 0 | 0 |
| | PC ← PC+1 t7 | Adr_ST+4番地 | 1 | 0 | 0 | 0 | 0 | × | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 |
| | | | : | : | : | : | : | : | : | : | : | : | : | : | : | : | : | : | : |
| $a_2$ | MAR ← PC t3 | Adr_AD番地 | 0 | 0 | 0 | 1 | 0 | × | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 0 | 0 |
| | MAR ← M[MAR] t4 | Adr_AD+1番地 | 0 | 0 | 0 | 0 | 0 | × | 0 | 0 | 0 | 0 | 1 | 0 | 1 | 1 | 0 | 0 | 0 |
| | AC ← AC+M[MAR] t5 | Adr_AD+2番地 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 0 |
| | PC ← PC+1 t6 | Adr_AD+3番地 | 1 | 0 | 0 | 0 | 0 | × | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 |
| | | | : | : | : | : | : | : | : | : | : | : | : | : | : | : | : | : | : |
| $a_3$ | MAR ← PC t3 | Adr_JZ番地 | 0 | 0 | 0 | 1 | 0 | × | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 0 | 0 |
| | PC ← PC+1 t4 | Adr_JZ+1番地 | 1 | 0 | 0 | 0 | 0 | × | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 |
| | Test(AC=0)? t5 | Adr_JZ+2番地 | 0 | 0 | 0 | 0 | 0 | × | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 1 |
| | PC ← M[MAR] t6 | Adr_JZ+3番地 | 0 | 0 | 1 | 0 | 0 | × | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 1 | 0 | 0 |

# 論理設計演習問題 2021 解答

(1) ST, AD, JZ 命令マイクロ操作列は？

| | LD | ST | AD | JZ |
|---|---|---|---|---|
| t3 | MAR ← PC | MAR ← PC | MAR ← PC | MAR ← PC |
| t4 | MAR ← M[MAR] | MAR ← M[MAR] | MAR ← M[MAR] | PC ← PC+1 |
| t5 | AC ← M[MAR] | MBR ← AC | AC ← AC+M[MAR] | Test(AC=0)? |
| t6 | PC ← PC+1 | M[MAR] ← MBR | PC ← PC+1 | PC ← M[MAR] |
| t7 | | PC ← PC+1 | | |

(2) μP-ROM の内容は?

| | | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| マイクロ操作列 | 番地 | IncPC | ClearPC | LoadPC | GPC | LoadIR | ALUMode | LoadAC | GAC | LoadMBR | GMBR | LoadMAR | R/W | GMemory | IncμPC | ClrμPC | LoadμPC | Tp |
| MAR ← PC | 0番地 | 0 | 0 | 0 | 1 | 0 | × | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 0 | 0 |
| IR ← M[MAR] | 1番地 | 0 | 0 | 0 | 0 | 1 | × | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 0 |
| PC ← PC+1 | 2番地 | 1 | 0 | 0 | 0 | 0 | × | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 |
| | | : | : | : | : | : | : | : | : | : | : | : | : | : | : | : | : | : |
| MAR ← PC | Adr_LD番地 | 0 | 0 | 0 | 1 | 0 | × | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 0 | 0 |
| MAR ← M[MAR] | Adr_LD+1番地 | 0 | 0 | 0 | 0 | 0 | × | 0 | 0 | 0 | 0 | 1 | 0 | 1 | 1 | 0 | 0 | 0 |
| AC ← M[MAR] | Adr_LD+2番地 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 0 |
| PC ← PC+1 | Adr_LD+3番地 | 1 | 0 | 0 | 0 | 0 | × | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 |
| | | : | : | : | : | : | : | : | : | : | : | : | : | : | : | : | : | : |
| MAR ← PC | Adr_ST番地 | 0 | 0 | 0 | 1 | 0 | × | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 0 | 0 |
| MAR ← M[MAR] | Adr_ST+1番地 | 0 | 0 | 0 | 0 | 0 | × | 0 | 0 | 0 | 0 | 1 | 0 | 1 | 1 | 0 | 0 | 0 |
| MBR ← AC | Adr_ST+2番地 | 0 | 0 | 0 | 0 | 0 | × | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 |
| M[MAR] ← MBR | Adr_ST+3番地 | 0 | 0 | 0 | 0 | 0 | × | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 1 | 0 | 0 | 0 |
| PC ← PC+1 | Adr_ST+4番地 | 1 | 0 | 0 | 0 | 0 | × | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 |
| | | : | : | : | : | : | : | : | : | : | : | : | : | : | : | : | : | : |
| MAR ← PC | Adr_AD番地 | 0 | 0 | 0 | 1 | 0 | × | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 0 | 0 |
| MAR ← M[MAR] | Adr_AD+1番地 | 0 | 0 | 0 | 0 | 0 | × | 0 | 0 | 0 | 0 | 1 | 0 | 1 | 1 | 0 | 0 | 0 |
| AC ← AC+M[MAR] | Adr_AD+2番地 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 0 |
| PC ← PC+1 | Adr_AD+3番地 | 1 | 0 | 0 | 0 | 0 | × | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 |
| | | : | : | : | : | : | : | : | : | : | : | : | : | : | : | : | : | : |
| MAR ← PC | Adr_JZ番地 | 0 | 0 | 0 | 1 | 0 | × | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 0 | 0 |
| PC ← PC+1 | Adr_JZ+1番地 | 1 | 0 | 0 | 0 | 0 | × | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 |
| Test(AC=0)? | Adr_JZ+2番地 | 0 | 0 | 0 | 0 | 0 | × | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 1 |
| PC ← M[MAR] | Adr_JZ+3番地 | 0 | 0 | 1 | 0 | 0 | × | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 1 | 0 | 0 |

(3) ClearμPC = φ(ClrμPC,Tp,Zero) = ClrμPC ∨ (Tp ∧ ¬Zero)

(4) IncPC = $t_2 \vee ((a_0 \vee a_2) \wedge t_6) \vee (a_1 \wedge t_7) \vee (a_3 \wedge t_4)$ LoadPC = $a_3 \wedge t_6$


# 第14章
# 乗算器と除算器

# 第14章 乗算器と除算器

- この章のねらい

## 14章 乗算器と除算器

乗算と除算は加減算とならんで基本的な数値演算である．本章では，固定小数点数を対象とする乗算器と除算器の構成方法について説明する．これらの演算器の実現方法には，逐次形のアルゴリズムにもとづく方法と並列形あるいはパイプライン形のアルゴリズムにもとづく方法がある．また，符号なし２進数と符号付き２進数の乗算および除算についても説明する．

# 表記法

- 2 進数のビット列表現
  - n ビットの 2 進数 X を $X = (x_{n-1},x_{n-2},\cdots,x_0)$ で表す.
- X の下位から（i+1）ビット目から（j +1）ビット目（i > j）までの部分列を X[i..j] で表す．すなわち
  - $X[i..j] = (x_i,\cdots,x_j),\ n>i>j \geq 0$
- ビット列の連接
  - 記号 & をビット列に対する連接（concatenetion）演算子として用いる．すなわち $X = (x_{n-1},x_{n-2},\cdots,x_0)$，$Y = (y_{m-1},y_{m-2},\cdots,y_0)$ に対して，X&Yを次のような列と定義する.
  - $X\&Y = (x_{n-1},x_{n-2},\cdots,x_0,y_{m-1},y_{m-2},\cdots,y_0)$
- ビット列とビットの論理積
  - X をビット列，y をビットとして，これらの論理積 $X\odot y$ を次式のように X の各ビットと y の論理積からなるビット列と定義する.
  - $X\odot y = (x_{n-1}\cdot y,x_{n-2}\cdot y,\cdots,x_0\cdot y)$

# 乗算器の種類

- 整数の 乗算（multiplication）は，二つの数値 X と Y の積 Z = X × Y を求め る演算である．乗算器（multiplier）は，乗算を実行する論理回路である．
  - 以下では X を 被乗数（multiplicand），Y を 乗数（multiplier），Z を 積（product）と呼ぶ．
  - 10 進法による乗算と 2 進法による乗算の例を 図14·1 に示す．2進数の乗算は（論理積と）加算とシフト演算の繰返しによって実現できる．

$$6 \times 7 = 42$$

$$\begin{array}{r} 6 \\ \times\ 7 \\ \hline 42 \end{array}$$

（a） 10 進法での乗算

$$0110 \times 0111 = 00101010$$

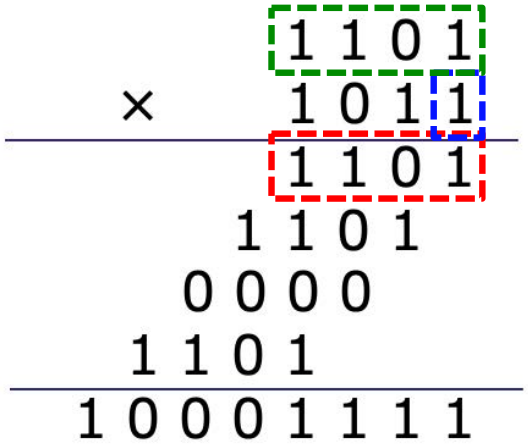
$$\begin{array}{r} 0\ 1\ 1\ 0 \\ \times\ \ 0\ 1\ 1\ 1 \\ \hline 0\ 1\ 1\ 0 \\ 0\ 1\ 1\ 0 \\ 0\ 1\ 1\ 0 \\ 0\ 0\ 0\ 0 \\ \hline 0\ 0\ 1\ 0\ 1\ 0\ 1\ 0 \end{array}$$

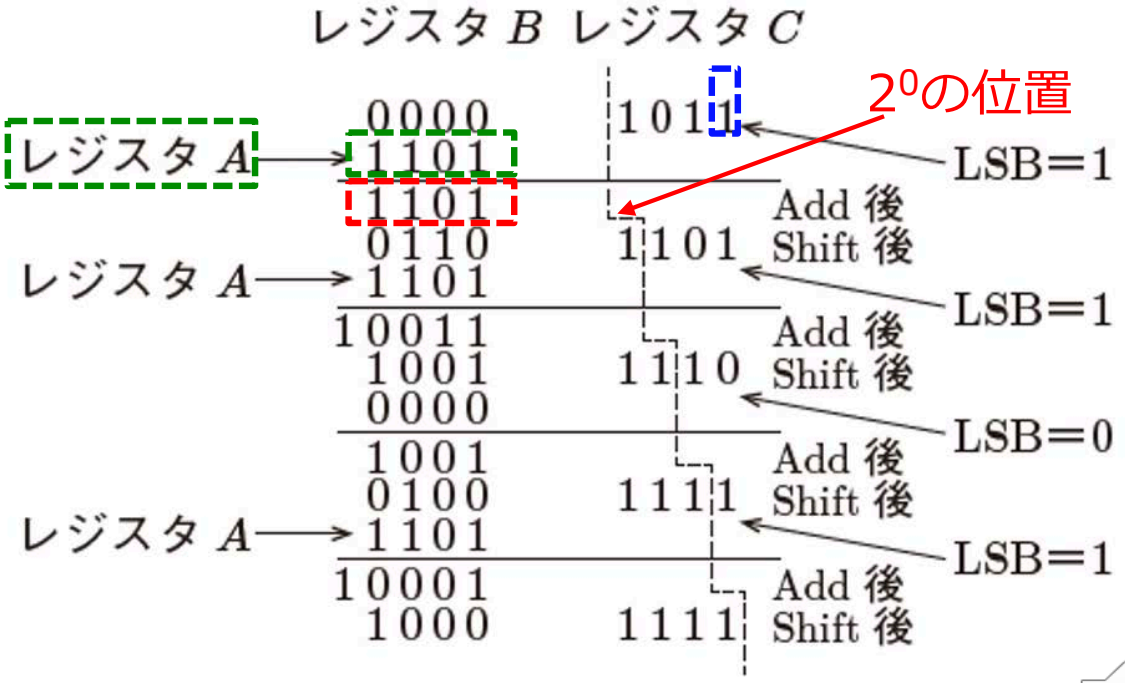
（b） 2 進法での乗算

**図 14・1** 10 進法および 2 進法での乗算の例

# 二つの整数の積を計算する逐次形乗算器

- 電卓の回路や自動販売機の制御回路と同様の方法で，二つの整数の積を計算する同期式順序回路を設計せよ
  - ２進数の掛け算（1101＊1011＝10001111＝13*11=143）

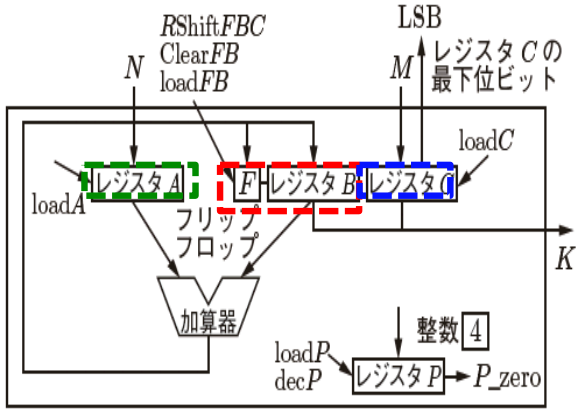



**解図 15・2** $N$=1101, $M$=1011 の積 $K$=10001111 を計算するアルゴリズムの概略

# 二つの整数の積を計算する逐次形乗算器

- 電卓の回路や自動販売機の制御回路と同様の方法で，二つの整数の積を計算する同期式順序回路を設計せよ
  - ２進数の掛け算（1101＊1011＝10001111＝13*11=143）



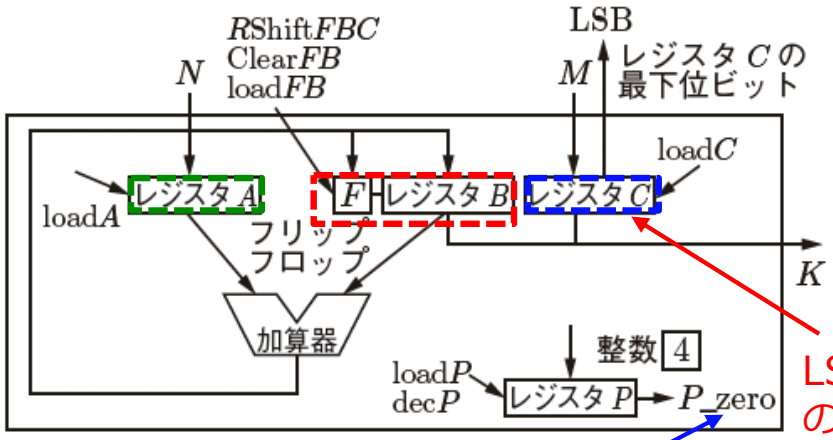

解図 15・1　非負整数 $N, M$ の積 $K$ を求める同期式順序回路

（イ）load$A$=1 のとき 外部入力 $N$ をレジスタ $A$ に格納する（$A$←input($N$) と書く）
load$A$=0 のとき レジスタ $A$ の値は不変
（ロ）R Shift$FBC$=1, Clear$FB$=0, load$FB$=0, load$C$=0 のとき
$F, B, C$ の値を 1 ビット右にシフトする（RShift($F, B, C$) と書く）
R Shift$FBC$=0, Clear$FB$=1, load$FB$=0 のとき $F, B$ の値を 0 にする（$\langle F, B\rangle$←0 と書く）
R Shift$FBC$=0, Clear$FB$=0, load$FB$=1 のとき 加算器の出力($A$+$B$)を$\langle F, B\rangle$に格納する
（$\langle F, B\rangle$←$A$+$B$ と書く）
R Shift$FBC$=0, load$C$=1 のとき
外部入力 $M$ をレジスタ $C$ に格納する（$C$←input($M$) と書く）
R Shift$FBC$=0, Clear$FB$=0, load$FB$=0, load$C$=0 のとき $F, B, C$ の値は不変
（ハ）load$P$=1, dec$P$=0 のとき 整数 4 をレジスタ $P$ に格納する（$P$←4 と書く）
load$P$=0, dec$P$=1 のとき レジスタ $P$ の値を 1 減らす（$P$←$P$−1 と書く）
load$P$=0, dec$P$=0 のとき レジスタ $P$ の値は不変
（ニ）$P$=0 が真 のときかつそのときのみ P_zero=1(test($P$=0)? と書く）
（ホ）レジスタ $C$ の最下位ピットが 1
のときかつそのときのみ LSB=1(test(LSB=1)? と書く）

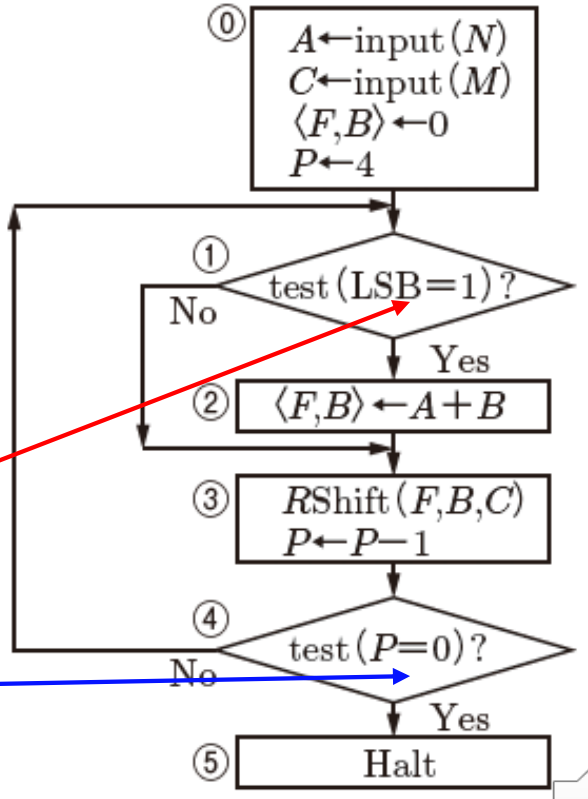
解図 15・3　各部品の動作仕様

# 二つの整数の積を計算する逐次形乗算器

- 電卓の回路や自動販売機の制御回路と同様の方法で，二つの整数の積を計算する同期式順序回路を設計せよ
  - ２進数の掛け算（1101＊1011＝10001111＝13*11=143）




解図 15・1　非負整数 $N$, $M$ の積 $K$ を求める同期式順序回路




解図 15・4　非負整数 $N, M$ の積 $K$ を求めるアルゴリズムのフローチャート

LSB:レジスタCの最下位ビット

P=0のとき、P__zero信号が 1 になる

# 逐次形乗算器

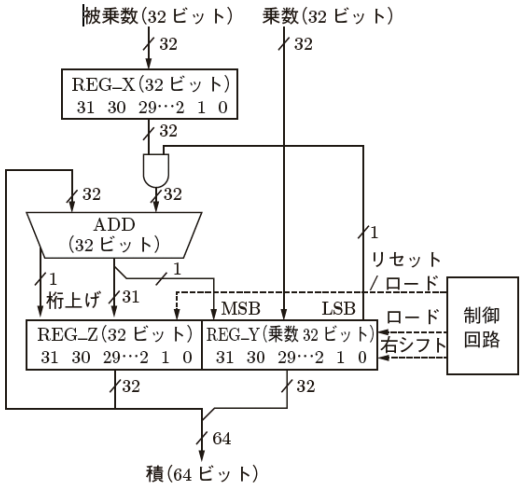



図 14・2 逐次形乗算器のブロック図

【32ビットの逐次形乗算アルゴリズム】

Step 1【初期化】

- レジスタ X およびレジスタ Y に，被乗数および乗数をそれぞれ格納する．レジスタ Z を 0 にリセットする．制御回路中のカウンタの値を 31 に初期化する．

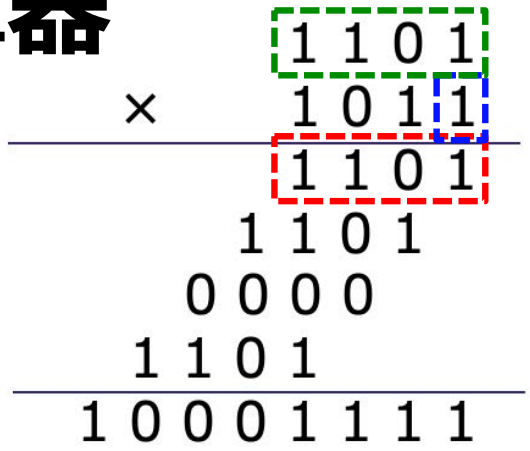
Step 2【加算とシフト】

1. Y [0] = 1 であればW ⇐ Z + X とし，そうでなければ W ⇐ Z とする．
2. レジスタ Y を右に 1 ビットシフトする．レジスタ Y の最上位ビットにはレジスタ W の最下位ビットをセットする．レジスタ Z には，レジスタ W の上位 32 ビットを格納する．

Step 3【終了判定】

1. カウンタの値が 0 であれば演算は終了．被乗数と乗数の積は，レジスタ Z とレジスタ Y の内容を結合して得られる 64 ビットの 2 進数である．
2. カウンタの値が 0 でなければ，カウンタから 1 を減じてStep 2 へ

# アレイ形乗算器




- 逐次形乗算器の計算手順の Step 2 および Step 3 の繰返しの処理を展開し，加算器を縦続接続して実行することである．このような方法で演算を実行する乗算器は アレイ形乗算器（array type maultipier）と呼ばれている．
  - アレイ形乗算器のブロック図を 図14·3 に示す．この図で $X \odot y_i$ はビット列 X とビット $y_i$ の論理積演算である．
- 図14·2 の乗算器は組合せ回路だけで構成されるので，乗算は 1 サイクルで実行可能である．ただし，この構成では（n−1）個の n ビット加算器を縦続接続しているので，回路全体の遅延時間は n ビット加算器の遅延時間の（n−1）倍になる．そのため，入力の桁数（n）が大きい場合には遅延時間が非常に長くなってしまう．

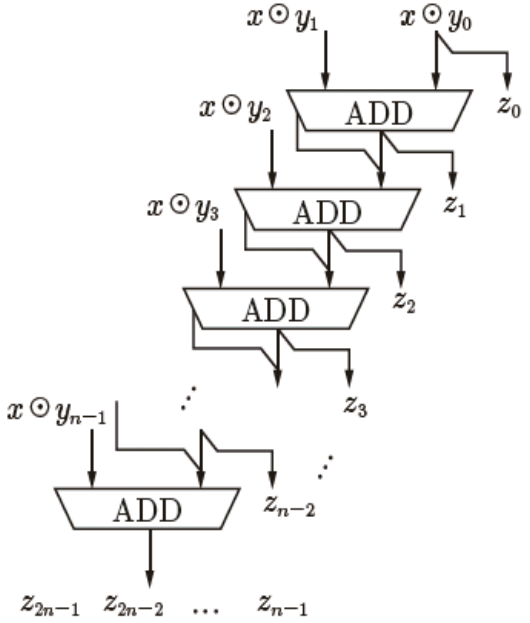



図 14・3 アレイ形乗算器のブロック図

# アレイ形乗算器

- 遅延時間をより効果的に短縮する方法として，桁上げ保存加算器（carry save adder）を用いる方法がある．n ビットの桁上げ保存加算器は，全加算器をn 個並列に並べただけの構成であり，回路の遅延時間は桁数にかかわらず一定（全加算器 1 個分の遅延時間と同じ）である．桁上げ保存加算器を用いた 6 ビットのアレイ形乗算器のブロック図を 図14·5 （一部間違い訂正有り）に示す．

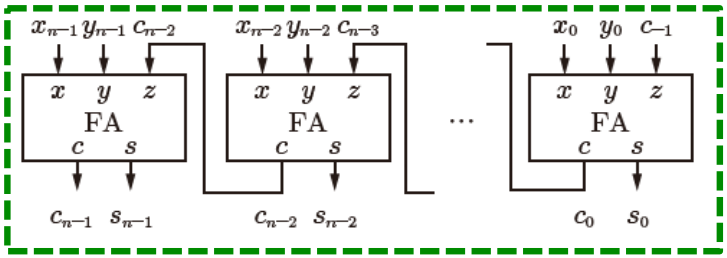
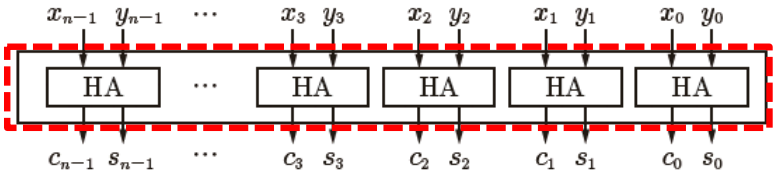
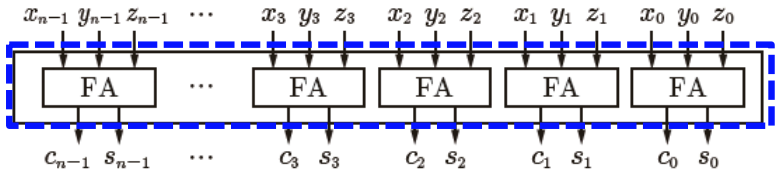



図 9・2 逐次桁上げ加算器




（a） 半加算器を用いた桁上げ保存加算器




（b） 全加算器を用いた桁上げ保存加算器

図 14・4 桁上げ保存加算器の構成

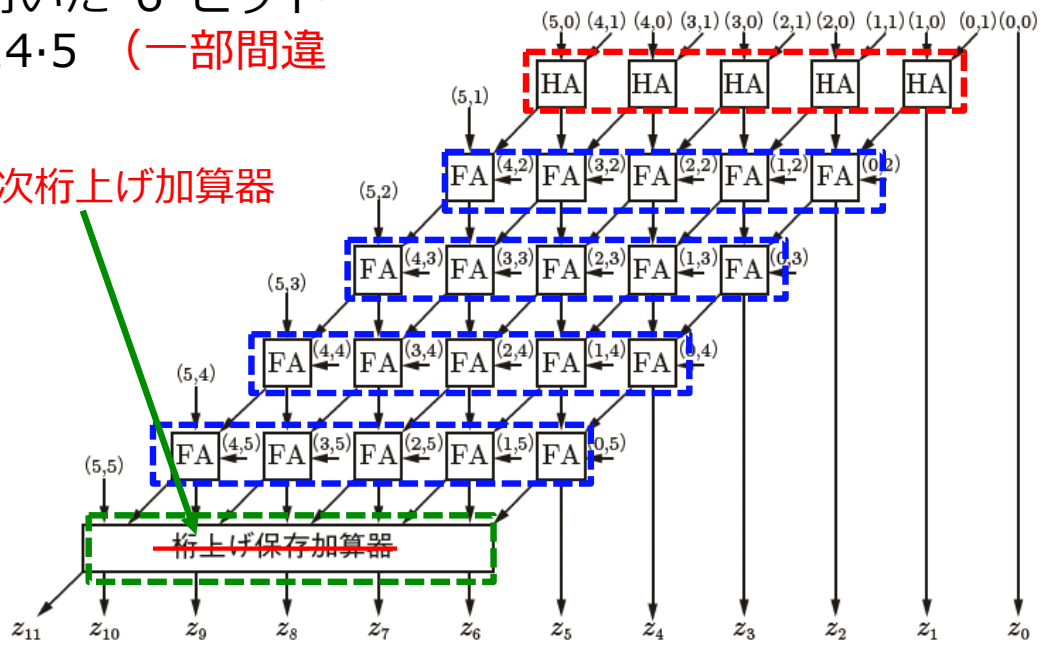



図 14・5 桁上げ保存加算器を用いたアレイ形乗算器のブロック

逐次桁上げ加算器と桁上げ保存加算器を組み合わせたアレイ型乗算器のブロック図

# アレイ形乗算器

- 遅延時間をより効果的に短縮する方法として，桁上げ保存加算器（carry save adder）を用いる方法がある．n ビットの桁上げ保存加算器は，全加算器をn 個並列に並べただけの構成であり，回路の遅延時間は桁数にかかわらず一定（全加算器 1 個分の遅延時間と同じ）である．桁上げ保存加算器を用いた 6 ビットのアレイ形乗算器のブロック図を 図14·5 （一部間違い訂正有り）に示す．

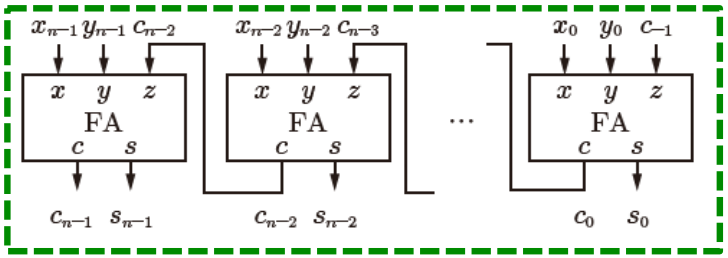
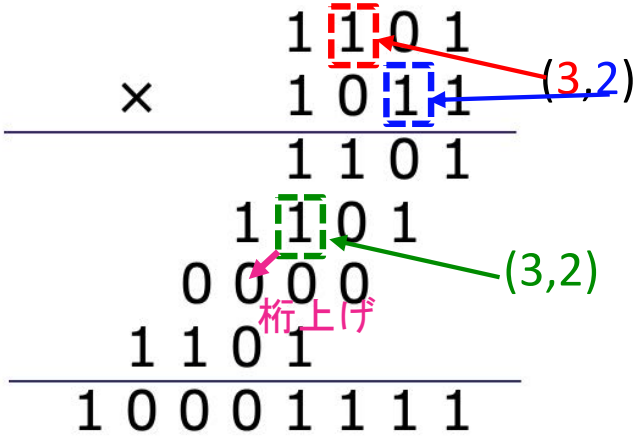



図 9・2 逐次桁上げ加算器



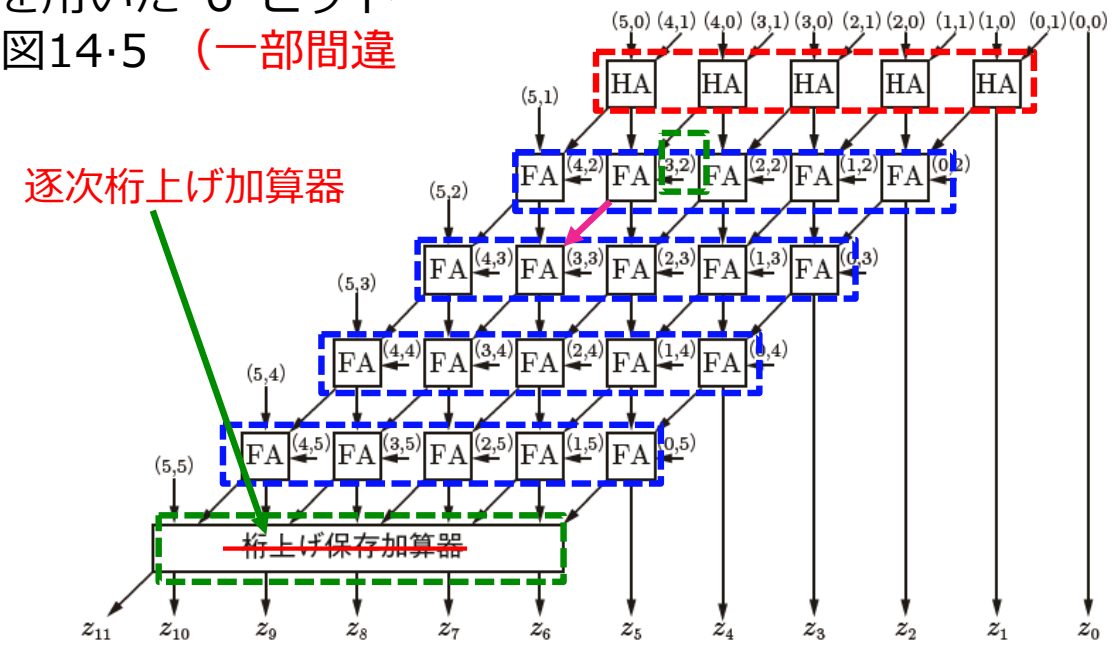




図 14・5 桁上げ保存加算器を用いたアレイ形乗算器のブロッ

逐次桁上げ加算器と桁上げ保存加算器を組み合わせたアレイ型乗算器のブロック図

# ツリー形乗算器

【ツリー形並列乗算アルゴリズム】

Step 1

- 次の式に従って次の四つの部分積を並列に計算する.
  - $Z_{7,6} \Leftarrow X \odot y_7 \times 2 + X \odot y_6$ (14・7)
  - $Z_{5,4} \Leftarrow X \odot y_5 \times 2 + X \odot y_4$ (14・8)
  - $Z_{3,2} \Leftarrow X \odot y_3 \times 2 + X \odot y_2$ (14・9)
  - $Z_{1,0} \Leftarrow X \odot y_1 \times 2 + X \odot y_0$ (14・10)

Step 2

赤字の部分誤植です

- 次の式に従って次の二つの部分積を並列に計算する.
  - $Z_{7,4} \Leftarrow Z_{7,6} \cdot 2^2 + Z_{5,4}$ (14・11)
  - $Z_{3,0} \Leftarrow Z_{3,2} \cdot 2^2 + Z_{1,0}$ (14・12)

Step 3

- 次の演算を実行する.
  - $Z \Leftarrow Z_{7,4} \cdot 2^4 + Z_{3,0}$ (14・13)

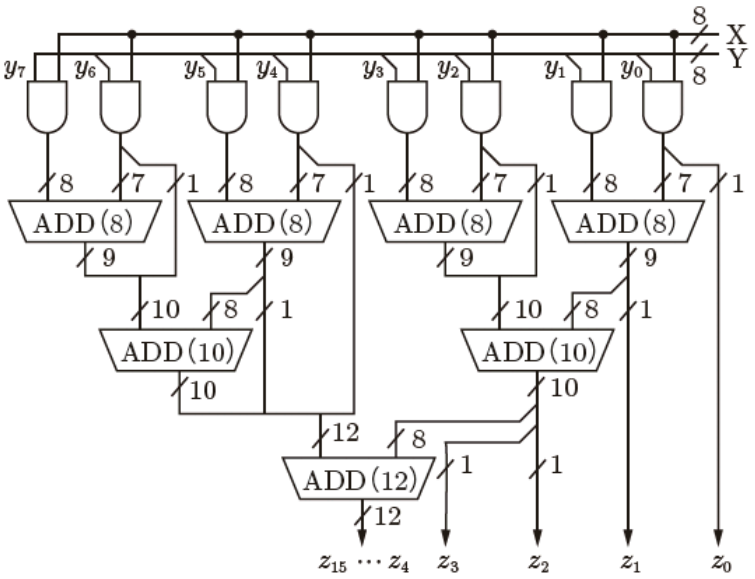



図 14・6 ツリー形乗算器のブロック図

# 符号付き数の乗算

- これまでは符号なし 2 進数の乗算方法について説明してきた．ここでは，符号付き 2 進数の乗算方法について説明する．
- 一般には，符号付き 2 進数の乗算は次のようにして実行する．
  - まず，被乗数および乗数の絶対値の乗算を行う．
  - 被乗数と乗数の符号が同じ（ともに正またはともに負）である場合には，絶対値の乗算結果をそのまま出力する．
  - 被乗数および乗数の符号が異なる場合（正と負または負と正）には，乗算結果は負になるので，絶対値の乗算結果の 2 の補数を出力する．

# 除算器

- 整数の除算（division）は，X を Y で割って得られる商 Q と剰余 R を求める演算である．除算器（divider）は，除算を実行する演算器である．以下では，X を被除数（dividend），Y を除数（divisor），Q を商（quotient），R を剰余（remainder）と呼ぶ．これらの数値の間には次の関係が成り立つ．
  - $X = Y \times Q + R$ (14・14)
  - $|R| < |Y|$ (14・15)
- 10 進法による除算と 2 進法による除算の例を 図14·7 に示す．この例に示すように，除算は減算とシフト演算の繰返しによって実現できる．

$13 \div 3 = 4 \cdots 1$

```
    4
3 ) 1 3
  1 2
  ---
    1
```

（a） 10 進法での除算

$1101 \div 11 = 100 \cdots 1$

```
      1 0 0
1 1 ) 1 1 0 1
      1 1
      -----
        0 0 1
        0 0
        -----
          0 1
          0 0
          -----
            1
```

（b） 2 進法での除算

**図 14・7** 引き戻し法にもとづく逐次形除算の実行例

# 引き戻し法にもとづく除算器

- 符号なし 2 進数の除算は 図14·7（b）に示すように，被徐数を 1 ビットずつシフトしながら被除数の一部と除数の比較を行い，選択的な減算を繰り返すことによって実現できる．この方法で32 ビットの被除数を32 ビットの除数で除算を行い，32 ビットの商と32 ビットの剰余を計算する除算器のブロック図を 図14·8 に示す．

13 ÷ 3 = 4 … 1

```
      4
  3 ) 1 3
      1 2
      ---
        1
```

（a） 10 進法での除算

1101 ÷ 11 = 100 … 1

```
        1 0 0
  1 1 ) 1 1 0 1
        1 1
        -----
          0 0 1
          0 0
          -----
            0 1
            0 0
            -----
              1
```

（b） 2 進法での除算

図 14・7 引き戻し法にもとづく逐次形除算の実行例

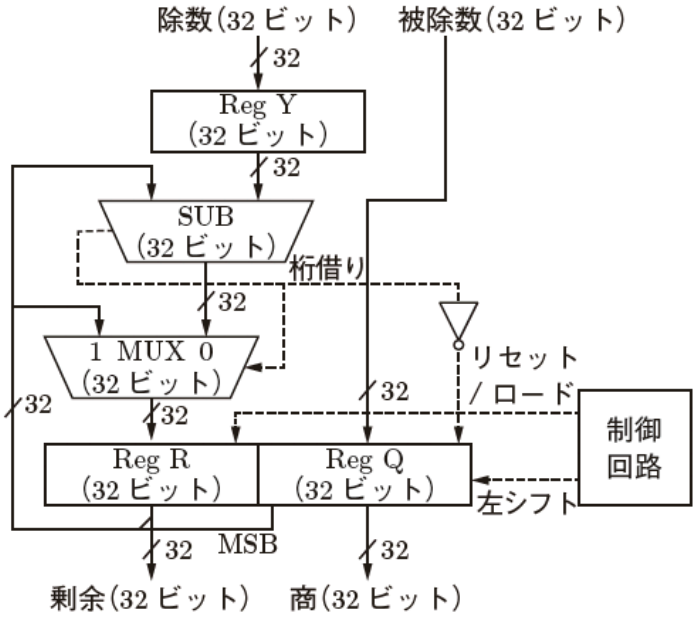



図 14・8 引き戻し法にもとづく逐次形除算器のブロック図

# 引き戻し法にもとづく除算器

Step 1【初期化】

- レジスタ Q およびレジスタ Y に，被除数および除数をそれぞれ格納する．レジスタR を 0 にリセットする．制御回路中のカウンタの値を 31 に初期化する．

Step 2【減算とシフト】

- W ⇐ R[30 .. 0]&Q[31] − Y とする．

1. 減算結果が正または 0（非負）（W[32] = 0）の場合には，R ⇐ W[31..0], Q ⇐Q[30..0]&1 とする．
2. そうでなければ，R ⇐ R[30..0]&Q[31], Q ⇐ Q[30..0]&0 とする．

Step 3【終了判定と剰余の補正】

1. カウンタの値が 0 であれば演算は終了．商および剰余はそれぞれ，レジスタQ およびレジスタR に格納されている．
2. カウンタの値が0 でなければ，カウンタの値を 1 減じて Step 2 へ．

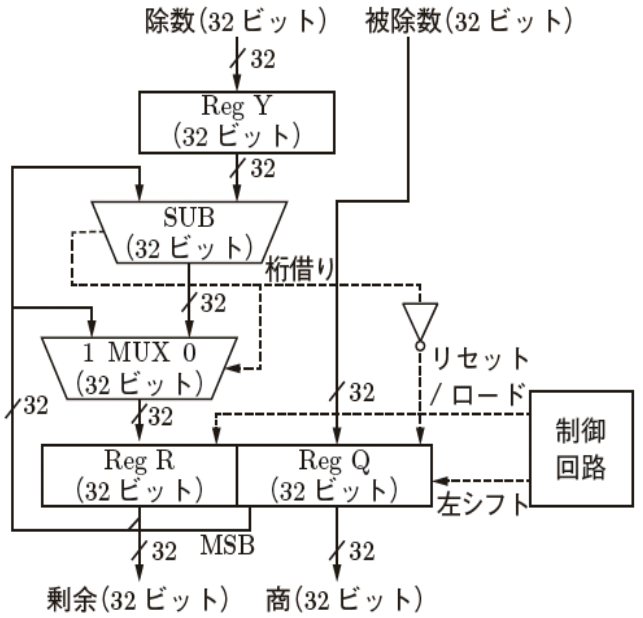



図 14・8 引き戻し法にもとづく逐次形除算器のブロック図

# 引き放し法にもとづく除算器

- 引き戻し法では，二つの数値を比較するために減算を実行し，その結果が負であった場合には，減算を実行する前の値を被除数に格納し直していた．この方法では，減算器の他に n ビットの 2 対 1 マルチプレクサが必要になり，ハードウェア量の増加と遅延時間の増加（動作周波数の低下）を招いてしまう．
- この節で紹介する方法は，減算結果が負になった場合に，すぐに被除数を減算前の値に復元せず，次の桁での処理の際に被除数を補正する方法である．そのために，この方法では減算器の代わりに加減算器を用いる．引き放し法による除算の実行例を 図14·9 に示す．引き放し法にもとづく除算器のブロック図を 図14·10 に示す．

1101 ÷ 11 = 100 … 1

```
                0 1 0 0     q=0100
0 0 1 1 ) 0 0 0 0 1 1 0 1
        −   0 0 1 1
          1 1 1 0 1         q(3)=0
        +     0 0 1 1
          0 0 0 0 0 0       q(2)=1
        −     0 0 1 1
            1 1 0 1 1       q(1)=0
        +     0 0 1 1
            1 1 1 1 0       q(0)=0
        +     0 0 1 1
            0 0 0 0 1       r=0001
```

図 14・9　引き放し法にもとづく除算の実行例

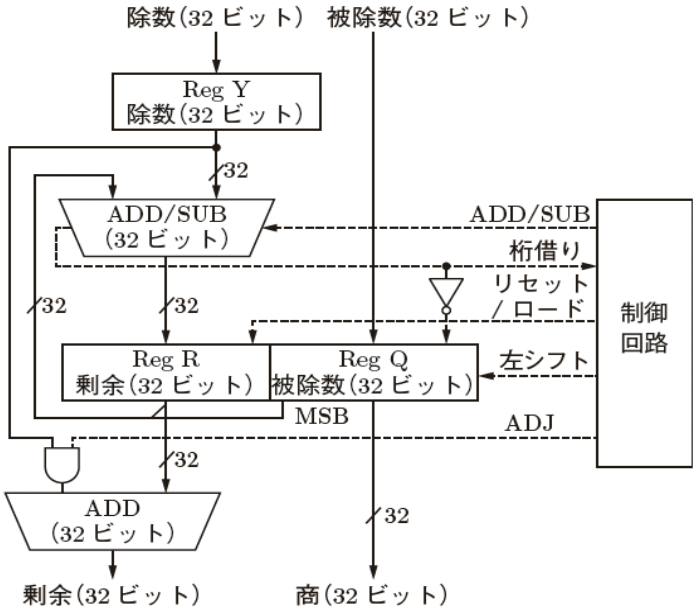



図 14・10　引き放し法にもとづく逐次形除算器のブロック図

# 引き放し法にもとづく除算器

【引き放し法にもとづく除算アルゴリズム】

Step 1【初期化】

- レジスタ Q およびレジスタ Y に，被除数および除数をそれぞれ格納する.
- レジスタ R およびレジスタ B を 0 にリセットする.
- 制御回路中のカウンタの値を 31 に初期化する.

- Step 2【減算または加算】
  1. B = 0 の場合，W ⇐ R[30..0]&Q[31] − Y
  2. B = 1 の場合，W ⇐ R[30..0]&Q[31] + Y
- Step 3【シフト】
  - B ⇐ W[32], R ⇐ W[31..0],
    Q ⇐ Q[30..0]&W[32] とする.

Step 4【終了判定】

1. カウンタの値が 0 であれば演算は終了．B = 1 であれば，R ⇐ R+Y とする．商および剰余はそれぞれ，レジスタ Q およびレジスタ R に格納されている.
2. カウンタの値が 0 でなければ，カウンタの値を 1 減じて Step 2 へ.

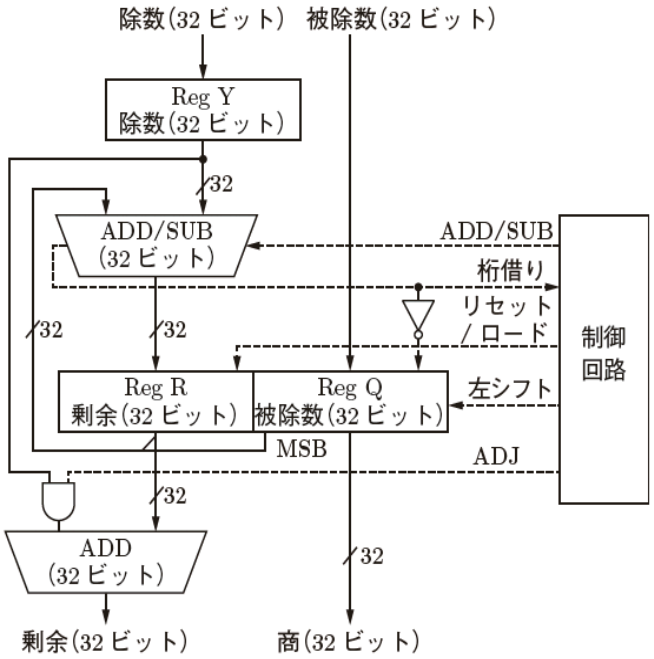



図 14・10 引き放し法にもとづく逐次形除算器のブロック図

## 演習問題

**1** 表 14·2 および表 14·3 に示す 4 進数の加算表および乗算表を完成させよ.

**表 14・2** 4 進数の加算表

| $x \setminus y$ | 0 | 1 | 2 | 3 |
|---|---|---|---|---|
| 0 | | | | |
| 1 | | | | |
| 2 | | | | |
| 3 | | | | |

**表 14・3** 4 進数の乗算表

| $x \setminus y$ | 0 | 1 | 2 | 3 |
|---|---|---|---|---|
| 0 | | | | |
| 1 | | | | |
| 2 | | | | |
| 3 | | | | |

**2** 非負の 4 進数 1 桁は 2 桁の 2 進数で表現できる．4 進数 $X = (x_1,\ x_0)$ および $Y = (y_1,\ y_0)$ の和は最大で 2 桁の 4 進数になる．$X$ と $Y$ の和を $Z = (z_3,\ z_2,\ z_1,\ z_0)$ とする．$z_3,\ z_2,\ z_1,\ z_0$ を $x_1,\ x_0,\ y_0,\ y_0$ を変数とする最も簡単な論理関数で表現せよ

**3** 非負の 4 進数 1 桁は 2 桁の 2 進数で表現できる．4 進数 $X = (x_1,\ x_0)$ および $Y = (y_1,\ y_0)$ の積は最大で 2 桁の 4 進数になる．$X$ と $Y$ の積を $Z = (z_3,\ z_2,\ z_1,\ z_0)$ とする．$z_3,\ z_2,\ z_1,\ z_0$ を $x_1,\ x_0,\ y_0,\ y_0$ を変数とする最も簡単な論理関数で表現せよ

**4** 次の 10 進数の組は被乗数および乗数を表している．これらの数値を 8 ビットの符号付き固定小数点形式で表現し，演算桁数を下位から 8 ビットに限定して乗算を行い，正しい結果が得られることを確認せよ.

(1) 3, 5
(2) 3, −5
(3) −3, 5
(4) −3, −5

# 演習問題 1

- 表14･2 および表14･3 に示す4 進数の加算表および乗算表を完成させよ.

表 14・2　4 進数の加算表

| $x \setminus y$ | 0 | 1 | 2 | 3 |
|---|---|---|---|---|
| 0 | | | | |
| 1 | | | | |
| 2 | | | | |
| 3 | | | | |

表 14・3　4 進数の乗算表

| $x \setminus y$ | 0 | 1 | 2 | 3 |
|---|---|---|---|---|
| 0 | | | | |
| 1 | | | | |
| 2 | | | | |
| 3 | | | | |

# 考慮時間

- 3分間程度で問題を解いてみてください．その間，ビデオを止めてください.

- この頁は30秒程度で次の頁に移行します.

# 演習問題１解答

- 表14･2 および表14･3 に示す4 進数の加算表および乗算表を完成させよ.

(解答)

- 右図参照

解表 14・1　4 進数の加算表

| $x \setminus y$ | 0 | 1 | 2 | 3 |
|---|---|---|---|---|
| 0 | 0 | 1 | 2 | 3 |
| 1 | 1 | 2 | 3 | 10 |
| 2 | 2 | 3 | 10 | 11 |
| 3 | 3 | 10 | 11 | 12 |

解表 14・2　4 進数の乗算表

| $x \setminus y$ | 0 | 1 | 2 | 3 |
|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 |
| 1 | 0 | 1 | 2 | 3 |
| 2 | 0 | 2 | 10 | 12 |
| 3 | 0 | 3 | 12 | 21 |

# 演習問題２

- 非負の 4 進数 1 桁は 2 桁の 2 進数で表現できる．4 進数 X = ($x_1$,$x_0$) および Y =($y_1$,$y_0$) の和は最大で 2 桁の 4 進数になる．X と Y の和を Z = ($z_3$,$z_2$,$z_1$,$z_0$)とする．$z_3$,$z_2$,$z_1$,$z_0$ を $x_1$,$x_0$,$y_1$,$y_0$ を変数とする最も簡単な論理関数で表現せよ.

（注意）

- 上記では， 4 進数 X を 2 ビットの 2 進数 ($x_1$,$x_0$) で表している.

# 考慮時間

- 5分間程度で問題を解いてみてください．その間，ビデオを止めてください.

- この頁は30秒程度で次の頁に移行します.

# 演習問題２解答

- 非負の 4 進数 1 桁は 2 桁の 2 進数で表現できる．4 進数 X = ($x_1$,$x_0$) および Y =($y_1$,$y_0$) の和は最大で 2 桁の 4 進数になる．X と Y の和を Z = ($z_3$,$z_2$,$z_1$,$z_0$)とする．$z_3$,$z_2$,$z_1$,$z_0$ を $x_1$,$x_0$,$y_1$,$y_0$ を変数とする最も簡単な論理関数で表現せよ.

（注意）

- 上記では，４進数 X を 2 ビットの 2 進数 ($x_1$,$x_0$) で表している.

（解答）

4進数なので，X,Yは共に0～3なので，
$z_3$=0
$z_2$ は X+Y が 4,5,6
$z_1$ は X+Y が 2,3,6
$z_0$ は 奇数

$$z_3 = 0$$

$$z_2 = x_1 \cdot y_1 \vee x_1 \cdot x_0 \cdot y_0 \vee x_0 \cdot y_1 \cdot y_0$$

$$z_1 = x_1 \cdot \overline{y}_1 \cdot \overline{y}_0 \vee x_1 \cdot \overline{x}_0 \cdot \overline{y}_1 \vee x_1 \cdot x_0 \cdot y_1 \cdot y_0 \vee \overline{x}_1 \cdot \overline{x}_0 \cdot y_1 \vee \overline{x}_1 \cdot y_1 \cdot \overline{y}_0 \vee \overline{x}_1 \cdot x_0 \cdot \overline{y}_1 \cdot y_0$$

$$z_0 = x_0 \cdot \overline{y}_0 \vee \overline{x}_0 \cdot y_0 = x_0 \oplus y_0$$

# 演習問題３

- 非負の 4 進数 1 桁は 2 桁の 2 進数で表現できる．4 進数 X = $(x_1,x_0)$ および Y =$(y_1,y_0)$ の積は最大で 2 桁の 4 進数になる．X と Y の和を Z = $(z_3,z_2,z_1,z_0)$とする．$z_3,z_2,z_1,z_0$ を $x_1,x_0,y_1,y_0$ を変数とする最も簡単な論理関数で表現せよ.

（注意）

- 上記では，４進数 X を 2 ビットの 2 進数 $(x_1,x_0)$ で表している.

# 考慮時間

- 5分間程度で問題を解いてみてください．その間，ビデオを止めてください.

- この頁は30秒程度で次の頁に移行します.

# 演習問題３解答

- 非負の 4 進数 1 桁は 2 桁の 2 進数で表現できる．4 進数 X = ($x_1$,$x_0$) および Y =($y_1$,$y_0$) の積は最大で 2 桁の 4 進数になる．X と Y の和を Z = ($z_3$,$z_2$,$z_1$,$z_0$)とする．$z_3$,$z_2$,$z_1$,$z_0$ を $x_1$,$x_0$,$y_1$,$y_0$ を変数とする最も簡単な論理関数で表現せよ．

（注意）

- 上記では，４進数 X を 2 ビットの 2 進数 ($x_1$,$x_0$) で表している．

（解答）

$$z_3 = x_1 \cdot x_0 \cdot y_1 \cdot y_0$$

$$z_2 = x_1 \cdot x_0 \cdot y_1 \cdot \overline{y}_0 \vee x_1 \cdot \overline{x}_0 \cdot y_1$$

$$z_1 = x_1 \cdot \overline{y}_1 \cdot y_0 \vee x_1 \cdot \overline{x}_0 \cdot y_0 \vee \overline{x}_1 \cdot x_0 \cdot \overline{y}_0$$

$$z_0 = x_0 \cdot y_0$$

4進数なので，X,Yは共に0～3なので，
$z_3$ = X × Y が 9
$z_2$ は X × Y が 4,6
$z_1$ は X × Y が 2,3,6
$z_0$ は X × Y が 1,3,9

解答
間違い

# 演習問題４

- 次の 10 進数の組は被乗数および乗数を表している．これらの数値を 8 ビットの符号付き固定小数点形式で表現し，演算桁数を下位から 8 ビットに限定して乗算を行い，正しい結果が得られることを確認せよ．

  (1) 3, 5

  (2) 3, −5

  (3) −3, 5

  (4) −3, −5

# 考慮時間

- 5分間程度で問題を解いてみてください．その間，ビデオを止めてください.

- この頁は30秒程度で次の頁に移行します.

# 演習問題４解答

- 次の 10 進数の組は被乗数および乗数を表している．これらの数値を 8 ビットの符号付き固定小数点形式で表現し，演算桁数を下位から 8 ビットに限定して乗算を行い，正しい結果が得られることを確認せよ．

  (1) 3, 5

  (2) 3, −5

  (3) −3, 5

  (4) −3, −5

（解答）

$$(1)\ 00000011 \times 00000101 = 00001111$$

$$(2)\ 00000011 \times 11111011 = 11110001$$

$$(3)\ 11111101 \times 00000101 = 11110001$$

$$(4)\ 11111101 \times 11111011 = 00001111$$

OSAKA UNIVERSITY

# 期末試験について

# 期末試験の実施方法について
## ～基本は中間試験と同じ～

- 体調不良などでこの日に受験できない場合は，事前にメールで連絡下さい．
- 試験は解答時間30分間＋CLEにアップするための時間10分の合計40分です．
- 1/27の15:00頃にZoomを立ち上げますので、15:10までに自身のPCのwebカメラをオンにしてログインして下さい．
  - Zoomのミーティング IDやパスコードは当日の９時にCLEに掲示します．
  - 15:10から試験の実施方法をZoomで説明します．試験は15:20開始，試験問題は15:20にCLE上で閲覧できるようになります．15:50頃を目安に解答を終了し，16:00までに解答をCLEにアップして下さい．学籍番号と名前を忘れず記載してください．
- 解答はスマホで写真を撮るか、スキャナでスキャンするなどの方法で電子化し，いつものレポート課題の提出と同じ方法でCLEにアップしてください．
- アップしてもらった解答がこちらでうまく印刷できない場合は，当日中にメールで連絡します．念の為，解答をメール送信できるように，翌日の28日まで保管しておいてください．
- 解答のファイルのアップに失敗した場合，higashino@ist.osaka-u.ac.jp までメールで解答を送信すると共に、そのことを上記のZoomのチャットで連絡ください．試験時間に質問がある場合もZoomのチャットで質問してください．
- 可能なら，CLEのお知らせに事前にアップした解答用紙のpdfを印刷してその用紙に解答を記入して下さい．難しい場合、自身のノートやレポート用紙に解答を記載しても構いません．いずれの場合も，読みやすい濃い字で答案を記載下さい．
- 配布したテキストやスライドの閲覧は自由です．

# 期末試験の試験範囲

## ～webカメラを準備下さい～

- 試験範囲
  - 論理回路の教科書の１２章～１５章＆付録（中間試験以降に授業した範囲）
- 理解してほしい項目
  - 順序回路の簡単化（状態数最小化）
  - Mealy型からMoore型，Moore型からMealy型への等価変換
  - かけ算回路の動作の仕組み
  - ICを用いた順序回路の設計法（Moore型順序回路としての実現）
  - マイクロプログラミング方式による実現
  - CPUの設計
- 試験の準備
  - 各章の章末の演習問題やレポート課題の内容を理解できるようにしておいてください．期末試験では，基本的に授業で説明した演習問題やレポート課題の内容と類似の問題を出題します．
- カメラの準備
  - 基礎工学部のオンライン試験では，試験中はwebカメラをオンにして試験を受けてくださいと通知が出ています．カメラを準備出来ない場合は，２０日までに東野まで連絡ください．別途指示します．


# 14回目の授業終了

# 授業終了

皆さん

今日はレポート課題はありません

# 授業終了

皆さん

期末試験の準備、よろしく！

来週は授業がありません.

試験は１月２７日（水）４限に実施します.