コンピュータアーキテクチャ（２）

坂井　修一
東京大学大学院 情報理工学系研究科 電子情報学専攻
東京大学　工学部　電子情報工学科／電気電子工学科

- はじめに
- データの流れと制御の流れ

はじめに

- 本講義の目的
  - コンピュータアーキテクチャの基本を学ぶ
- 時間・場所
  - 火曜日 8:30 – 10:15　工学部2号館241
- ホームページ（ダウンロード可能）
  - url:　http://www.mtl.t.u-tokyo.ac.jp/~sakai/hard/
- 教科書
  - **坂井修一『コンピュータアーキテクチャ』（コロナ社、電子情報レクチャーシリーズC-9）**
  - **坂井修一『実践　コンピュータアーキテクチャ』（コロナ社）**

  教科書通りやります
- 参考書
  - D. Patterson and J. Hennessy, Computer Organization & Design、3rd Ed.（邦訳『コンピュータの構成と設計』(第3版)上下（日系ＢＰ））
  - 馬場敬信『コンピュータアーキテクチャ』（改訂２版）、オーム社
  - 富田眞治『コンピュータアーキテクチャⅠ』、丸善
- 予備知識：　論理回路
  - 坂井修一『論理回路入門』、培風館
- 成績
  - 試験＋レポート＋出席

講義の概要と予定（１／２）

１．コンピュータアーキテクチャ入門

ディジタルな表現、負の数、実数、加算器、ＡＬＵ，フリップフロップ、レジスタ、計算のサイクル

２．　データの流れと制御の流れ

主記憶装置、メモリの構成と分類、レジスタファイル、命令、命令実行の仕組み、実行サイクル、算術論理演算命令、シーケンサ、条件分岐命令

３．命令セットアーキテクチャ

操作とオペランド、命令の表現形式、アセンブリ言語、命令セット、
算術論理演算命令、データ移動命令、分岐命令、アドレシング、
サブルーチン、RISCとCISC

４．パイプライン処理（１）

パイプラインの原理、命令パイプライン、オーバヘッド、構造ハザード、データハザード、制御ハザード

５．パイプライン処理（２）

フォワーディング、遅延分岐、分岐予測、命令スケジューリング

６．キャッシュ

記憶階層と局所性、透過性、キャッシュ、ライトスルーとライトバック、
ダイレクトマップ型、フルアソシアティブ型、セットアソシアティブ型、キャッシュミス

7．仮想記憶

仮想記憶、ページフォールト、TLB、物理アドレスキャッシュ、仮想アドレスキャッシュ、メモリアクセス機構

講義の概要と予定（２／２）

８．基本CPUの設計

ディジタル回路の入力、Verilog HDL、シミュレーションによる動作検証、アセンブラ、基本プロセッサの設計、基本プロセッサのシミュレーションによる検証

９．命令レベル並列処理（１）

並列処理、並列処理パイプライン、VLIW、スーパスカラ、並列処理とハザード

１０．命令レベル並列処理（２）

静的最適化、ループアンローリング、ソフトウェアパイプライニング、トレーススケジューリング

１１．アウトオブオーダ処理

インオーダーとアウトオブオーダー、フロー依存、逆依存、出力依存、
命令ウィンドウ、リザベーションステーション、レジスタリネーミング、
マッピングテーブル、リオーダバッファ、プロセッサの性能

１２．入出力と周辺装置

周辺装置、ディスプレイ、二次記憶装置、ハードウェアインタフェース、割り込みとポーリング、アービタ、DMA、例外処理

試験：　7月

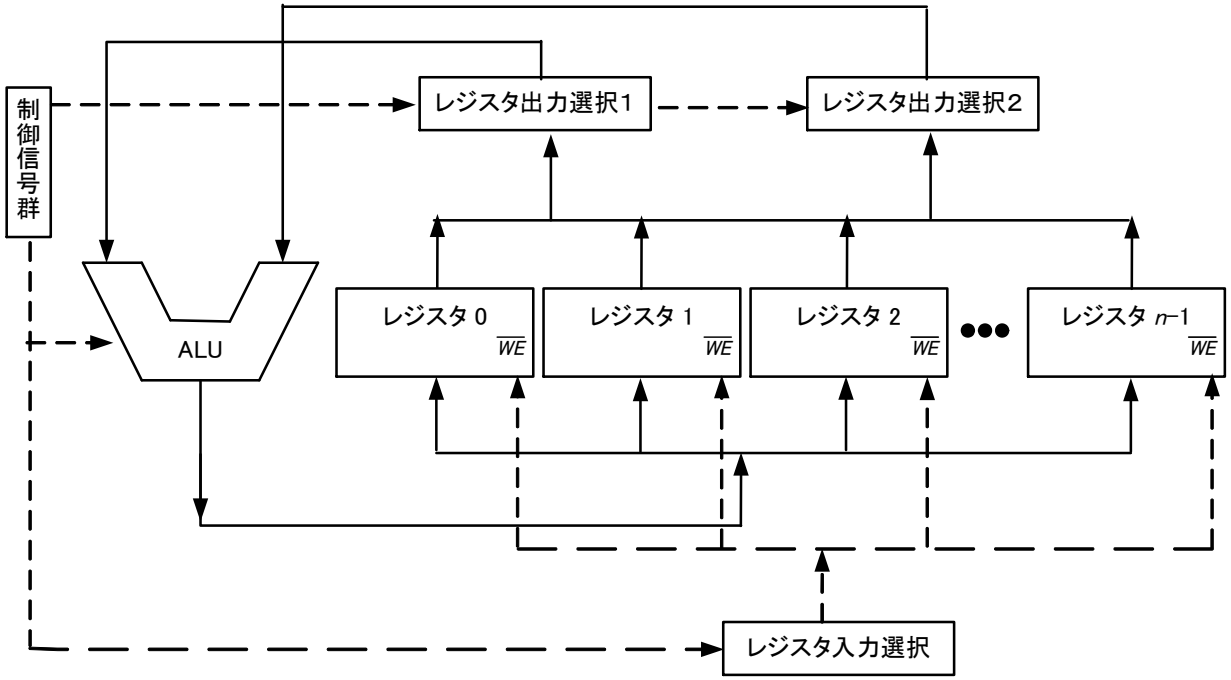
１．前回のまとめ

- 2進法によるディジタルな表現
  - n本の線でn桁の2進数を表現する
- 2の補数表示
  - 2^n-xで$-x$を表す。
- 実数の表現
  - 固定小数点と浮動小数点方式がある
- 組み合わせ回路による演算回路
  - 加算器、減算器、加減算器、ALU（算術論理演算ユニット）
- フリップフロップ
  - 1ビットの記憶回路でD,JKフリップフロップなどがある
- レジスタ
  - フリップフロップを並列にnビット並べたもの
- 演算のサイクル
  - レジスタ＝＞ALU=>レジスタの繰り返し

２．データの流れと制御の流れ

- 内容
  - 主記憶装置
    - レジスタ＋ALU？
    - 主記憶装置
    - メモリの構成
    - メモリの分類
    - レジスタファイル
  - 命令
    - 命令とはなにか
    - 命令実行の仕組み
    - 算術論理演算命令の実行サイクル
    - メモリ操作命令の実行サイクル
  - シーケンサ
    - シーケンサとはなにか
    - 条件分岐命令の実行サイクル
  - 練習問題

「レジスタ＋ALU」だけでいいか？



- 疑問
  - レジスタ群だけ大量のデータが扱えるか？
  - 制御信号はどうやって作られるのだろうか？
  - 「条件*P*が満足されれば*A*を実行し、満足されなければ*B*を実行する」（条件分岐）
    「*C*を100回繰り返す」（繰り返し実行）はどうやって実現するのか。

主記憶装置

- 大容量記憶の実現

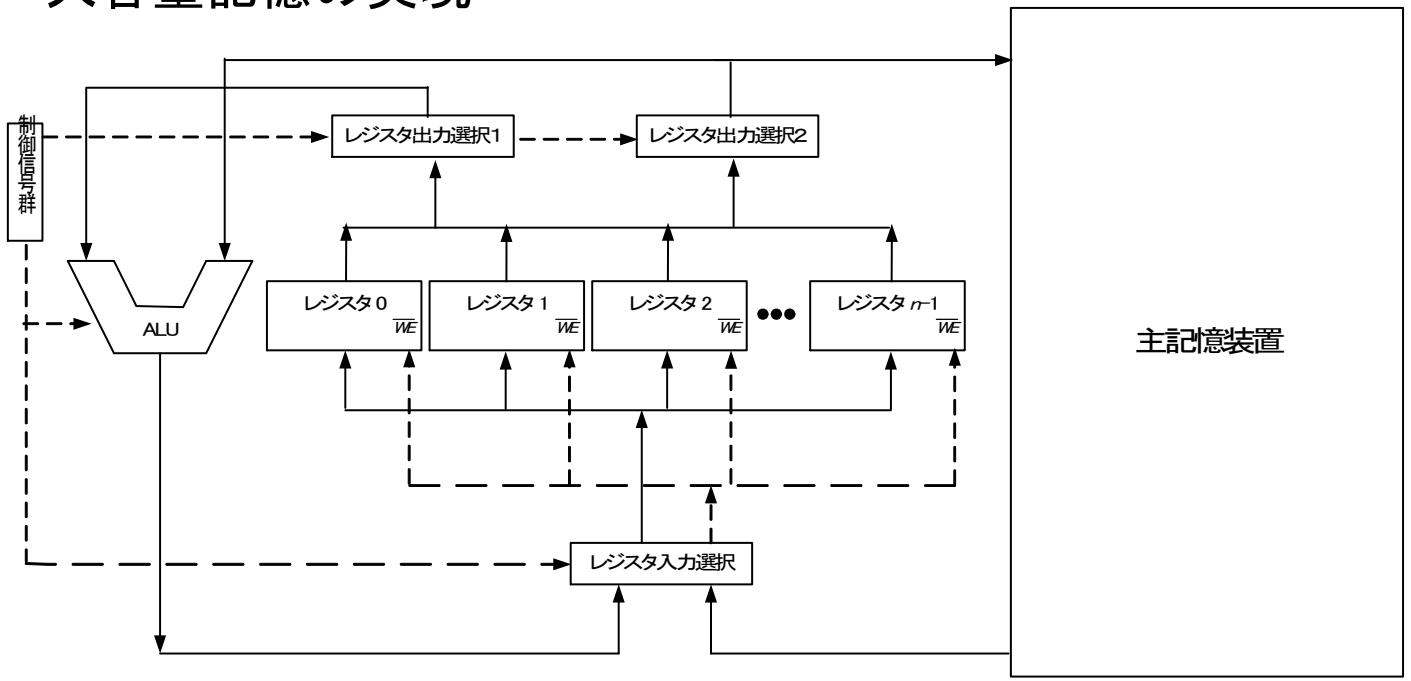


図 2.1　主記憶装置を含む演算実行機構

主記憶装置を含む演算のサイクル

1) 主記憶装置からレジスタにデータを移動させる（読み出し、read）
2) レジスタ⇒ＡＬＵ⇒レジスタ（計算、calculation)
3) レジスタから主記憶装置にデータを移動させる（書き込み、write)

アドレスデコーダ

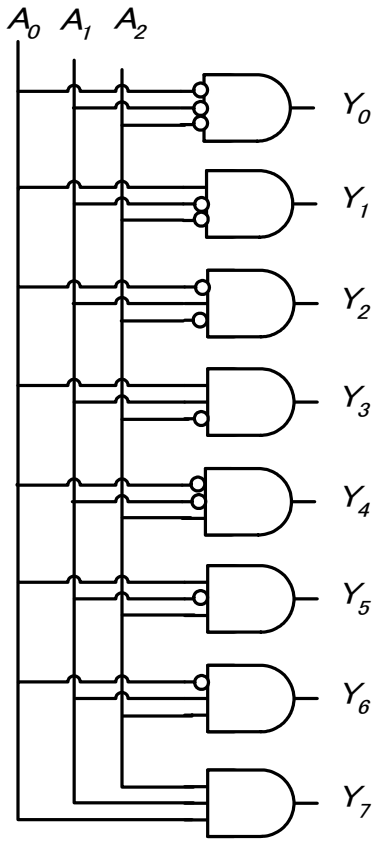


図 2.3　３入力８出力デコーダ

©Shuichi Sakai

メモ　©Toshiyuki Nakata

nビットのアドレスで2^n個の回路の内の1個を選ぶ。左図は3ビットのメモリで8個のYの内の1個を選ぶ

アドレスデコーダ

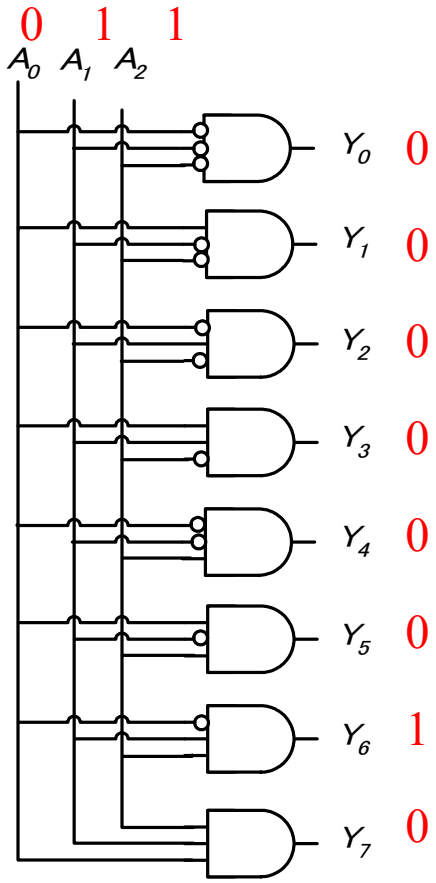


図 2.3　３入力８出力デコーダ

©Shuichi Sakai

メモ ©Toshiyuki Nakata

nビットのアドレスで2^n個の回路の内の1個を選ぶ。左図は3ビットのメモリで8個のYの内の1個を選ぶ

アドレスデコーダ

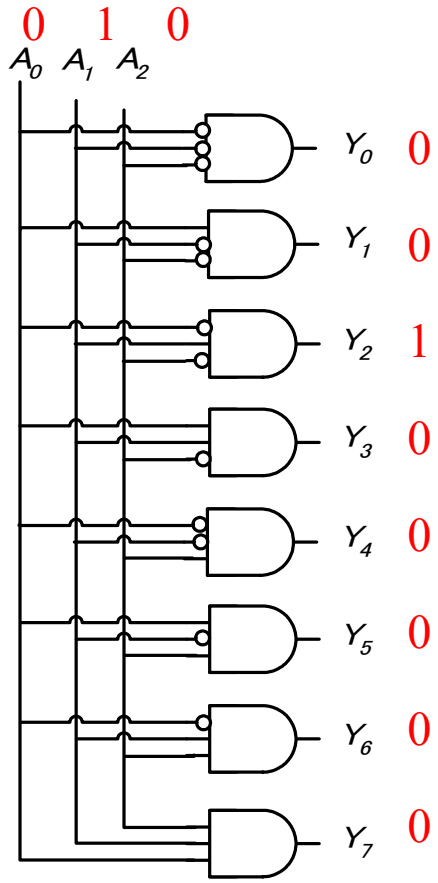


図 2.3　３入力８出力デコーダ

©Shuichi Sakai

メモ ©Toshiyuki Nakata

nビットのアドレスで2^n個の回路の内の1個を選ぶ。左図は3ビットのメモリで8個のYの内の1個を選ぶ

メモリの構成

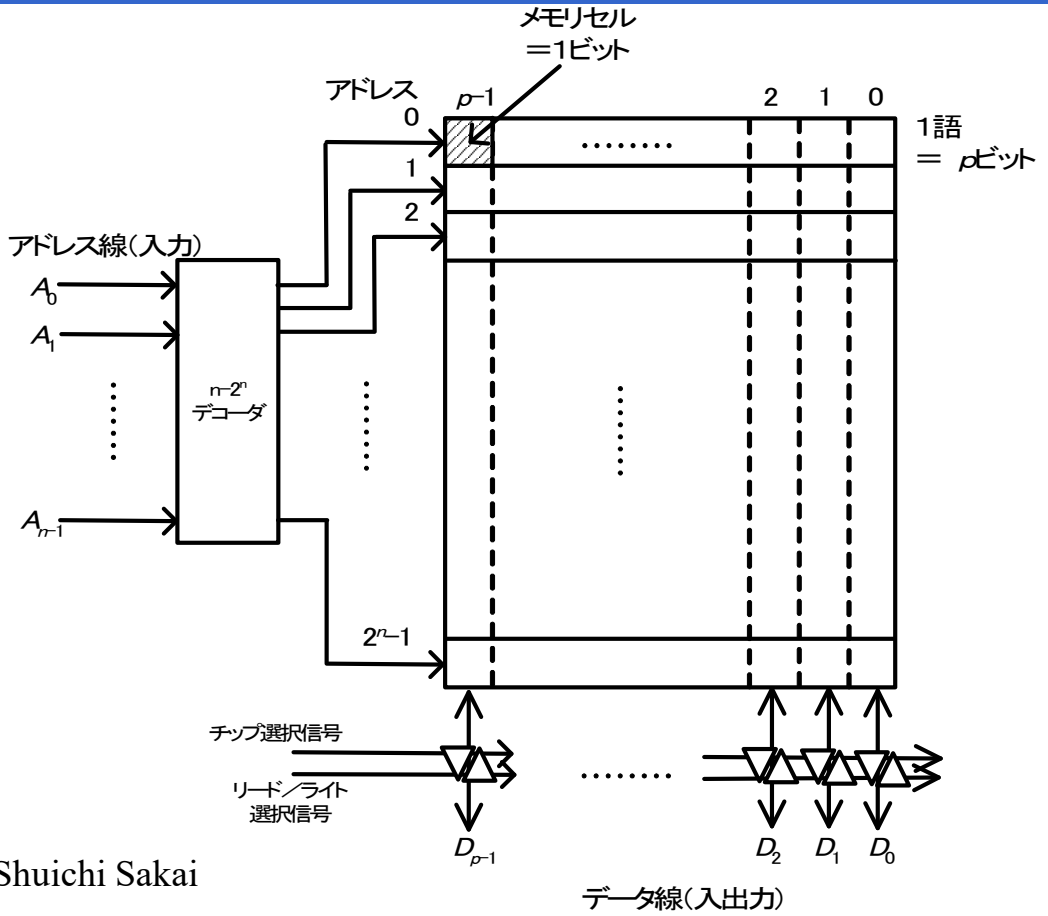
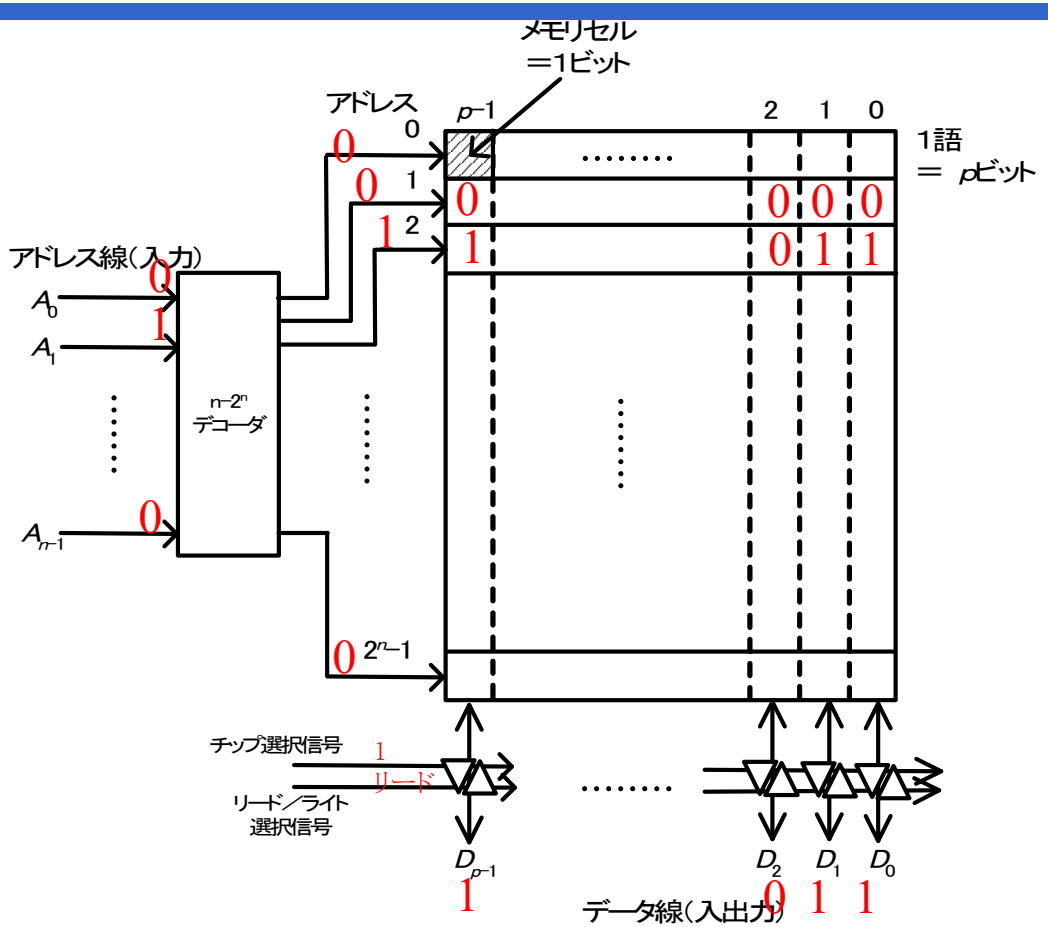


図 2.2　メモリの構成

(1) リード（read、読み出し）：与えられたアドレスに記憶されているデータを読み出す
(2) ライト (write、書き込み)：与えられたアドレスに与えられたデータを書き込む。

メモリの構成



©Shuichi Sakai　　図 2.2　メモリの構成

(1) リード（read、読み出し）：与えられたアドレスに記憶されているデータを読み出す.

2番地のデータを読む場合

メモリの構成

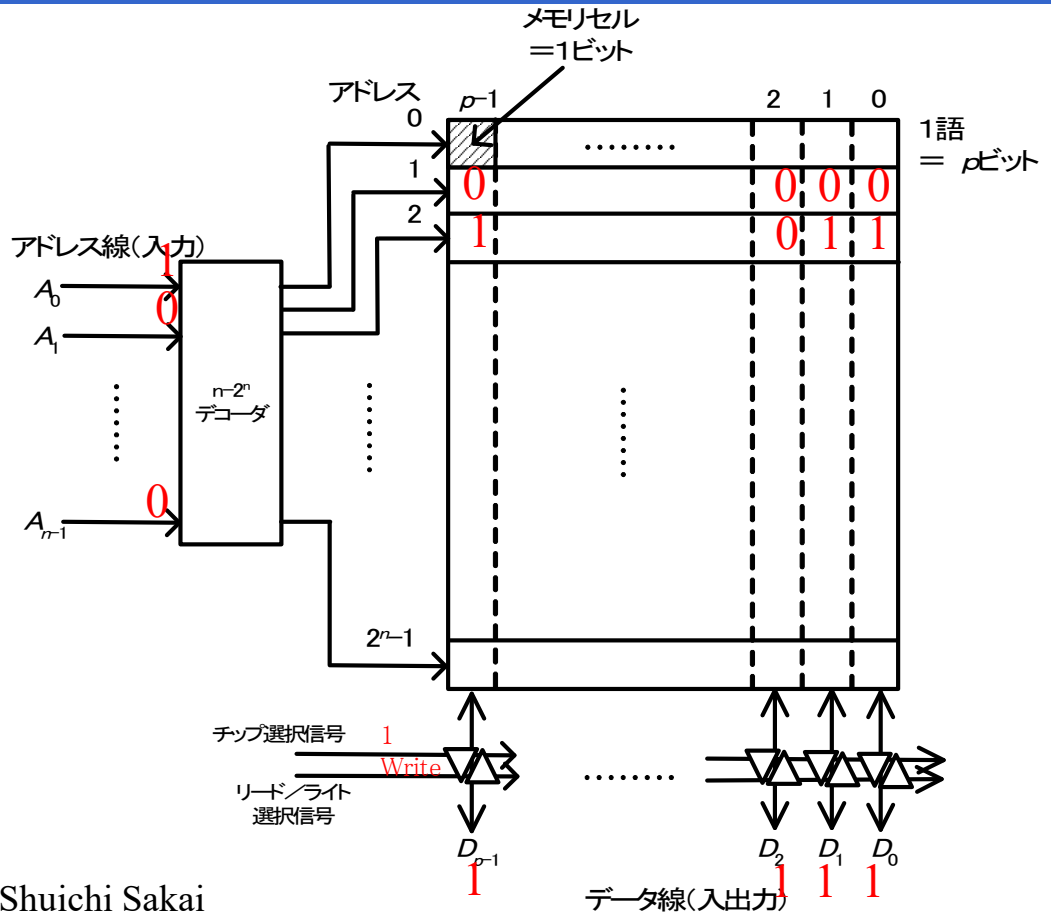


図 2.2　メモリの構成

(2) ライト (write、書き込み) : 与えられたアドレスに与えられたデータを書き込む

1番地にデータを書き込む場合

メモリの構成

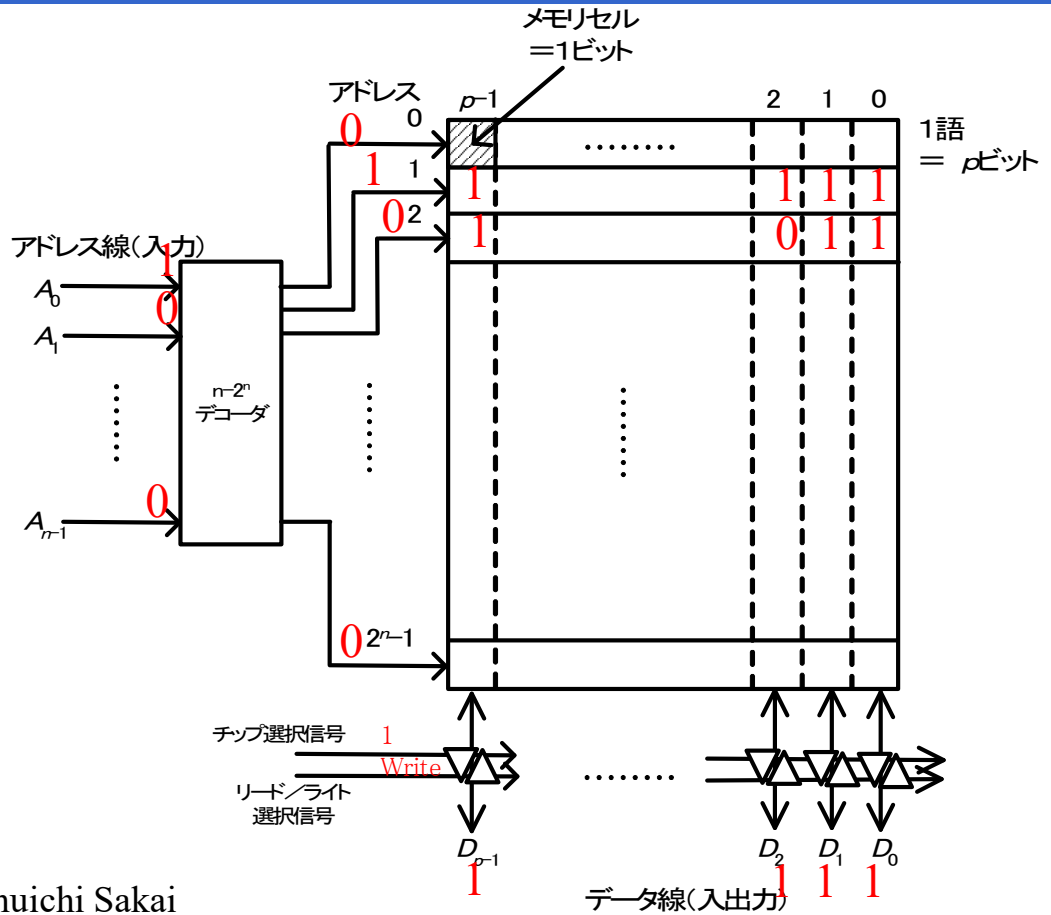
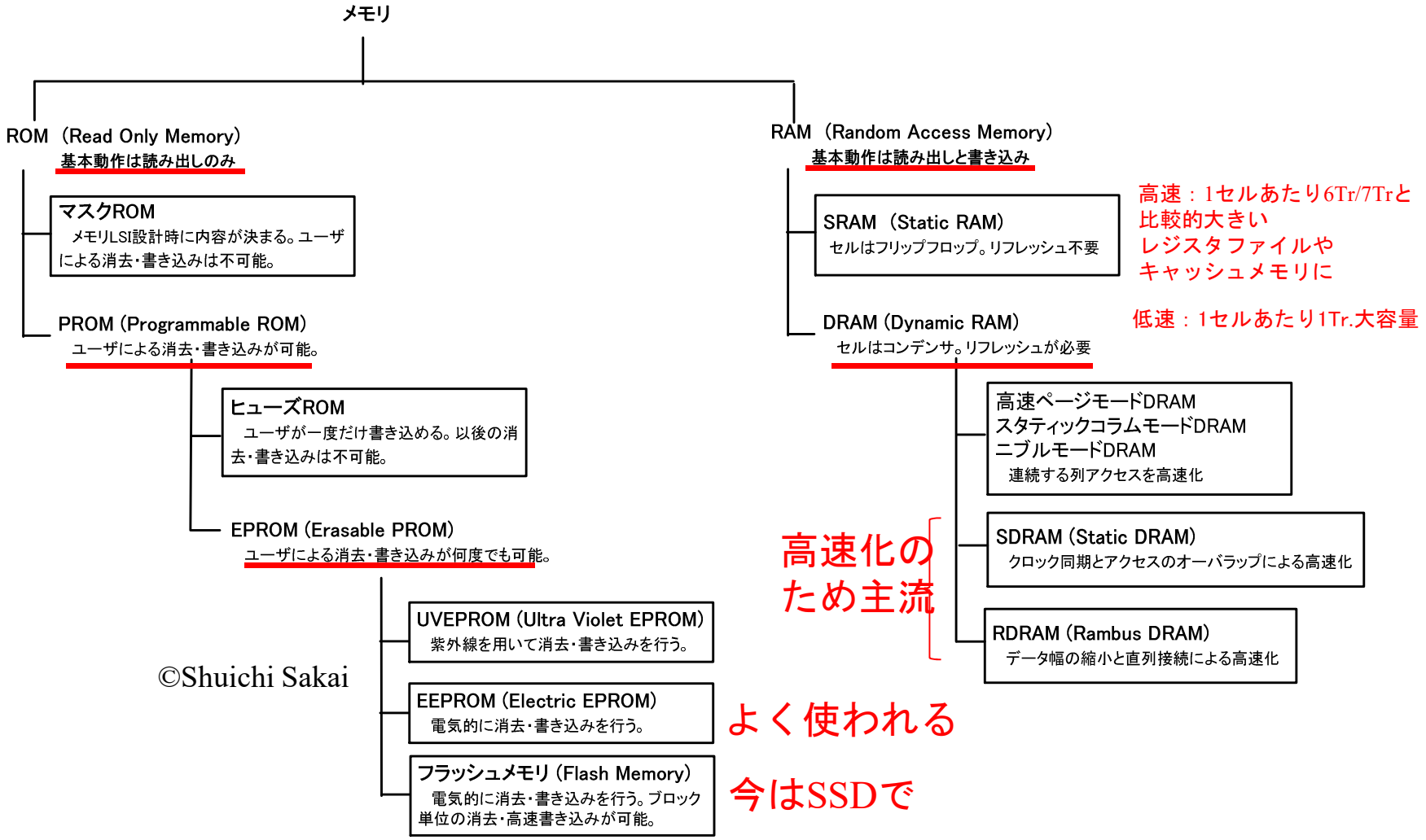


図 2.2　メモリの構成

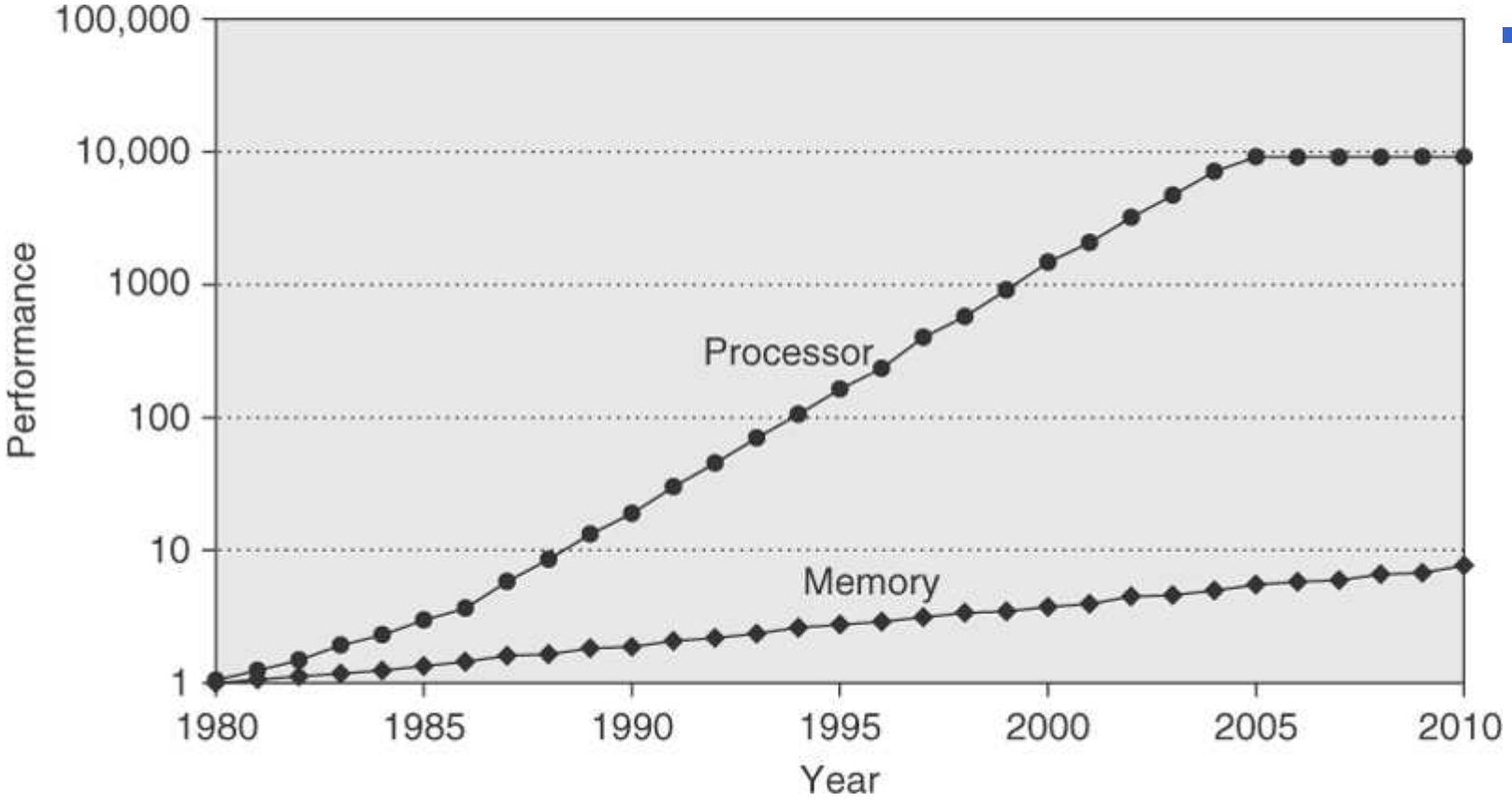
(2) ライト (write、書き込み) : 与えられたアドレスに与えられたデータを書き込む

1番地にデータを書き込む場合

メモリの分類



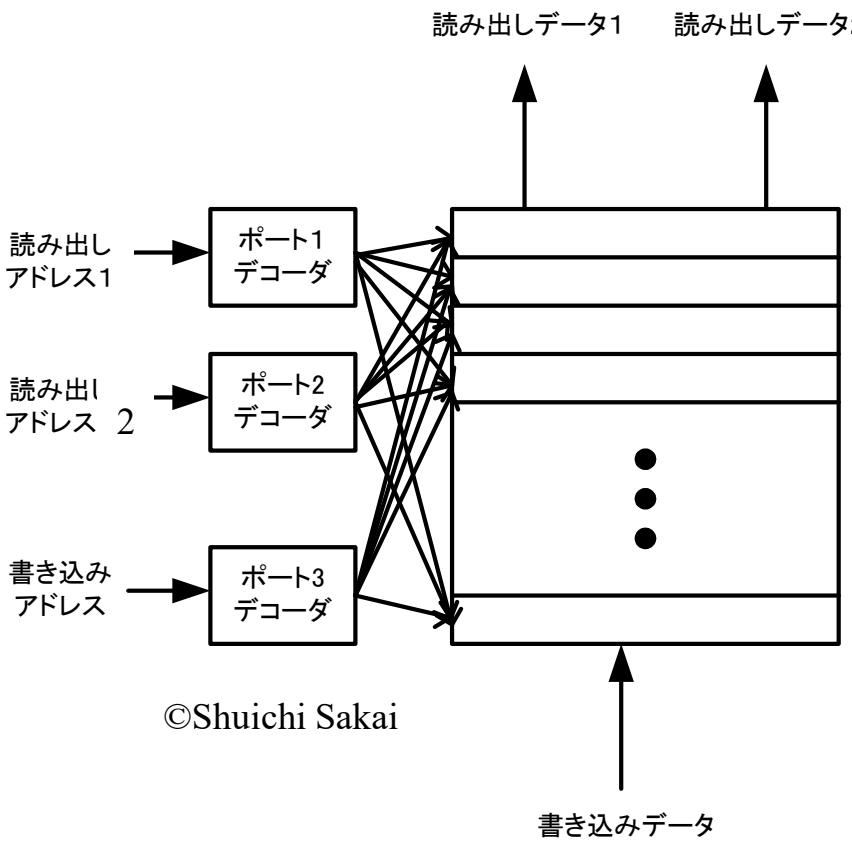
DRAMメモリのLatencyの深刻さ（メモ）

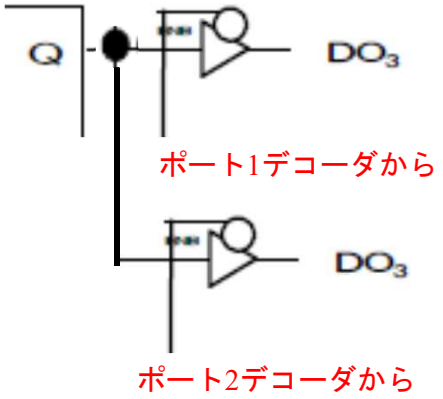


30年間の間に、プロセッサのLatencyは1/10000になったのに、
メモリのlatencyは1/10にしかなっていない。それを解決するために、
キャッシュメモリやメモリからのデータをブロックで転送するDDRAMな
使われるようになった。

Hennessy & Patterson：Computer Architecture A Quantitative Approach pp.73

レジスタファイル

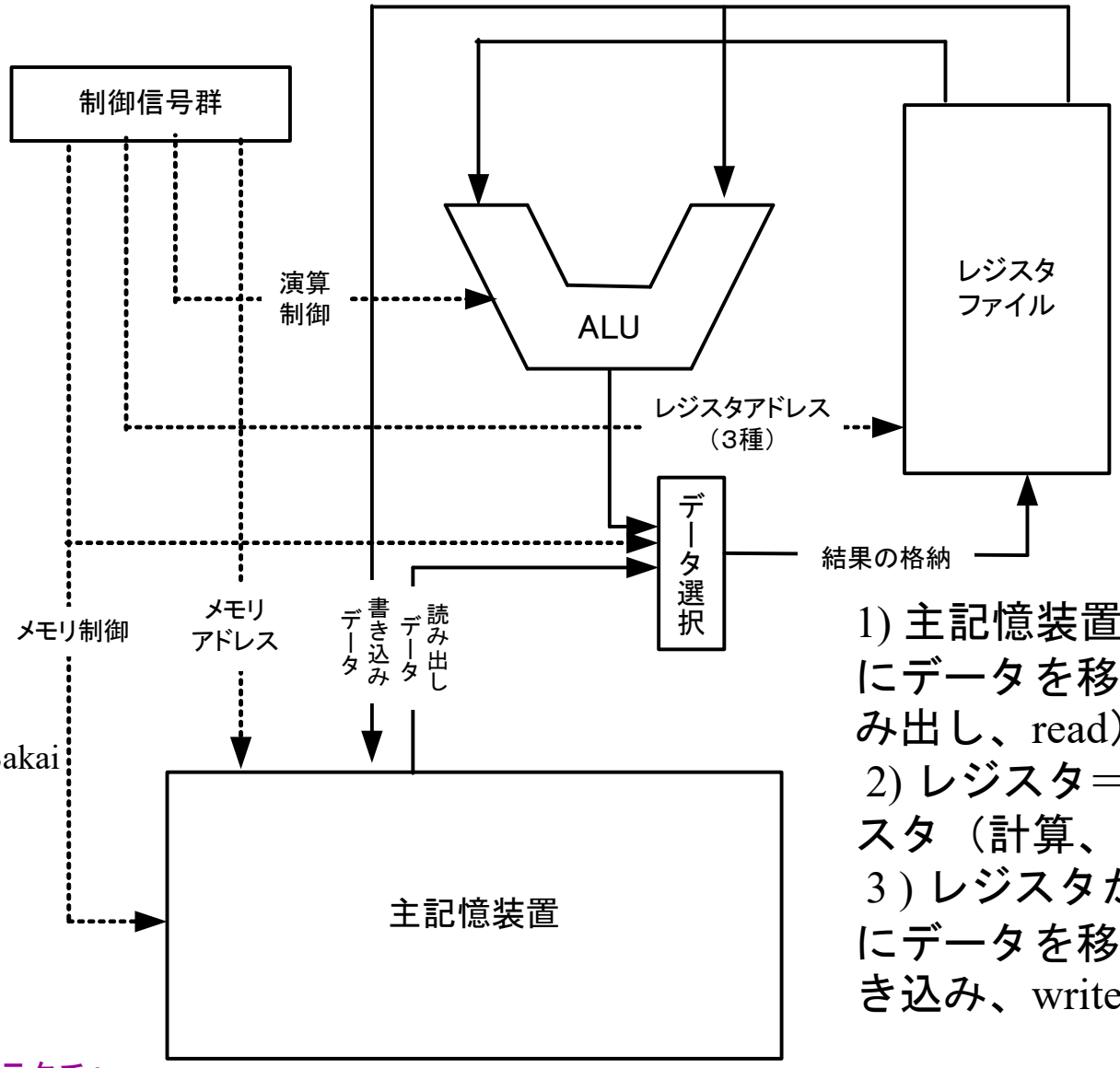




2出力の1つの実装方法

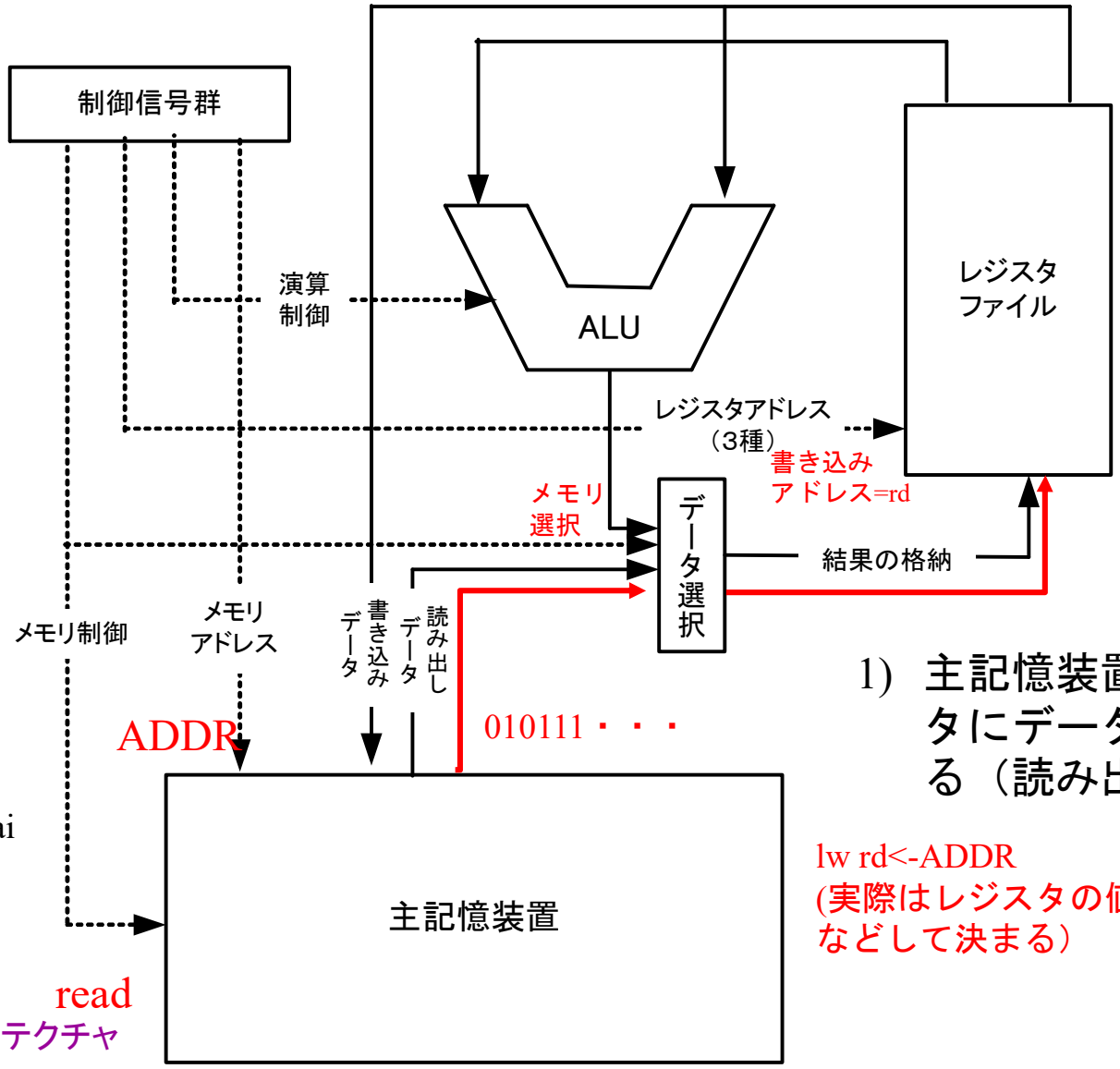
- 1入力2出力（3ポート）
- アドレス
- SRAM
- 典型的には32個のレジスタからなる

主記憶装置の接続



1) 主記憶装置からレジスタにデータを移動させる（読み出し、read）
2) レジスタ⇒ＡＬＵ⇒レジスタ（計算、calculation)
3) レジスタから主記憶装置にデータを移動させる（書き込み、write)

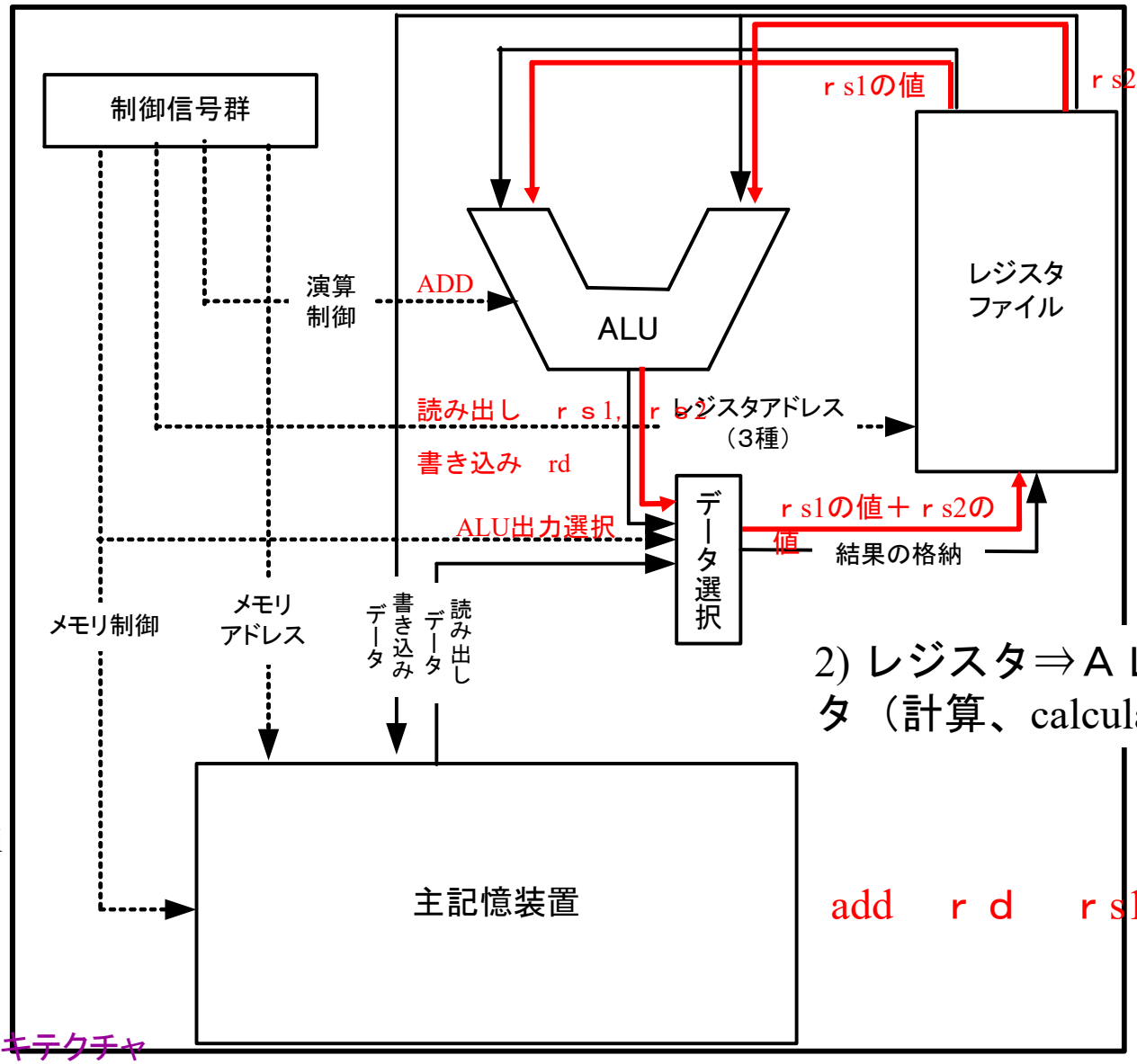
主記憶装置の接続



1) 主記憶装置からレジスタにデータを移動させる（読み出し、read）

lw rd<-ADDR
(実際はレジスタの値と加算などして決まる)

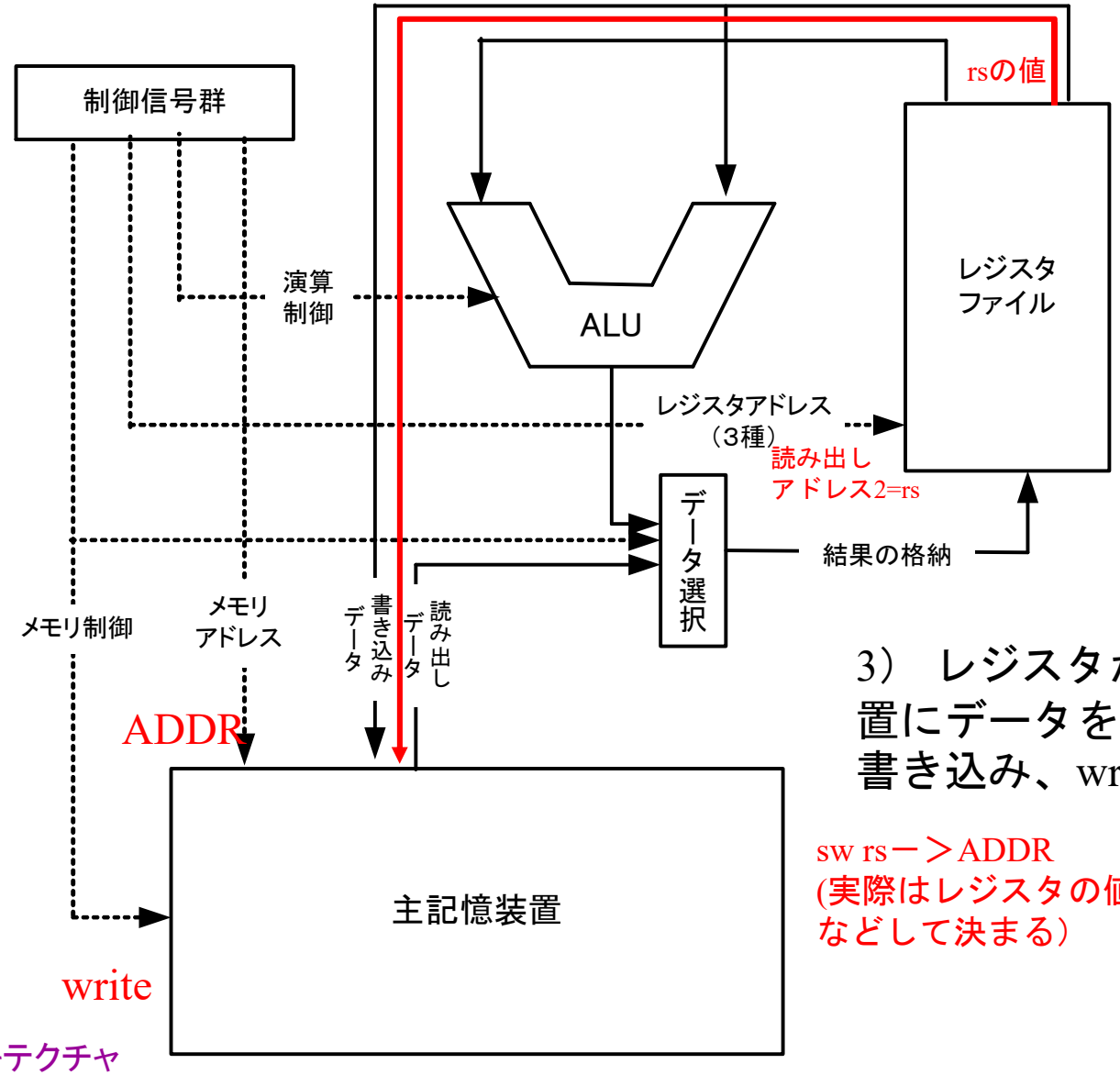
主記憶装置の接続



2) レジスタ⇒ＡＬＵ⇒レジスタ（計算、calculation)

add　ｒｄ　ｒs1,rs2

主記憶装置の接続

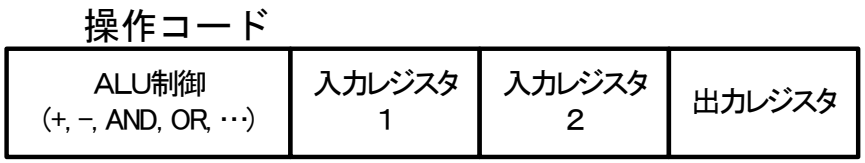


3） レジスタから主記憶装置にデータを移動させる（書き込み、write)

sw rs－＞ADDR
(実際はレジスタの値と加算などして決まる)

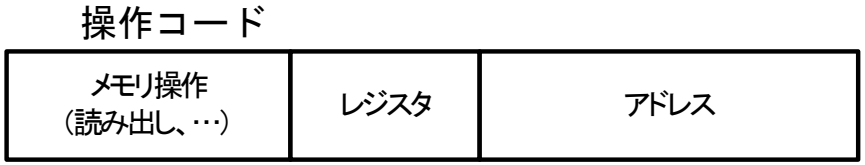
命令

- 命令
  - コンピュータの動作を指定するもの
  - 「データ」として表現される
    - コンピュータの最大の発明＝von Neumann Type
- 命令の構成
  - 数個のフィールドから成る
    - 操作コード
    - レジスタ番地
    - メモリ番地
    - 実行細則
- 命令の格納
  - メモリに入れられる
  - 読み出して使われる



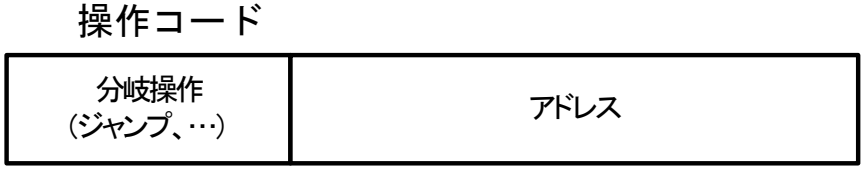
出力レジスタ ← 入力レジスタ1 op 入力レジスタ2

(1) 算術論理演算命令



レジスタ ← メモリの「アドレス」番地の内容

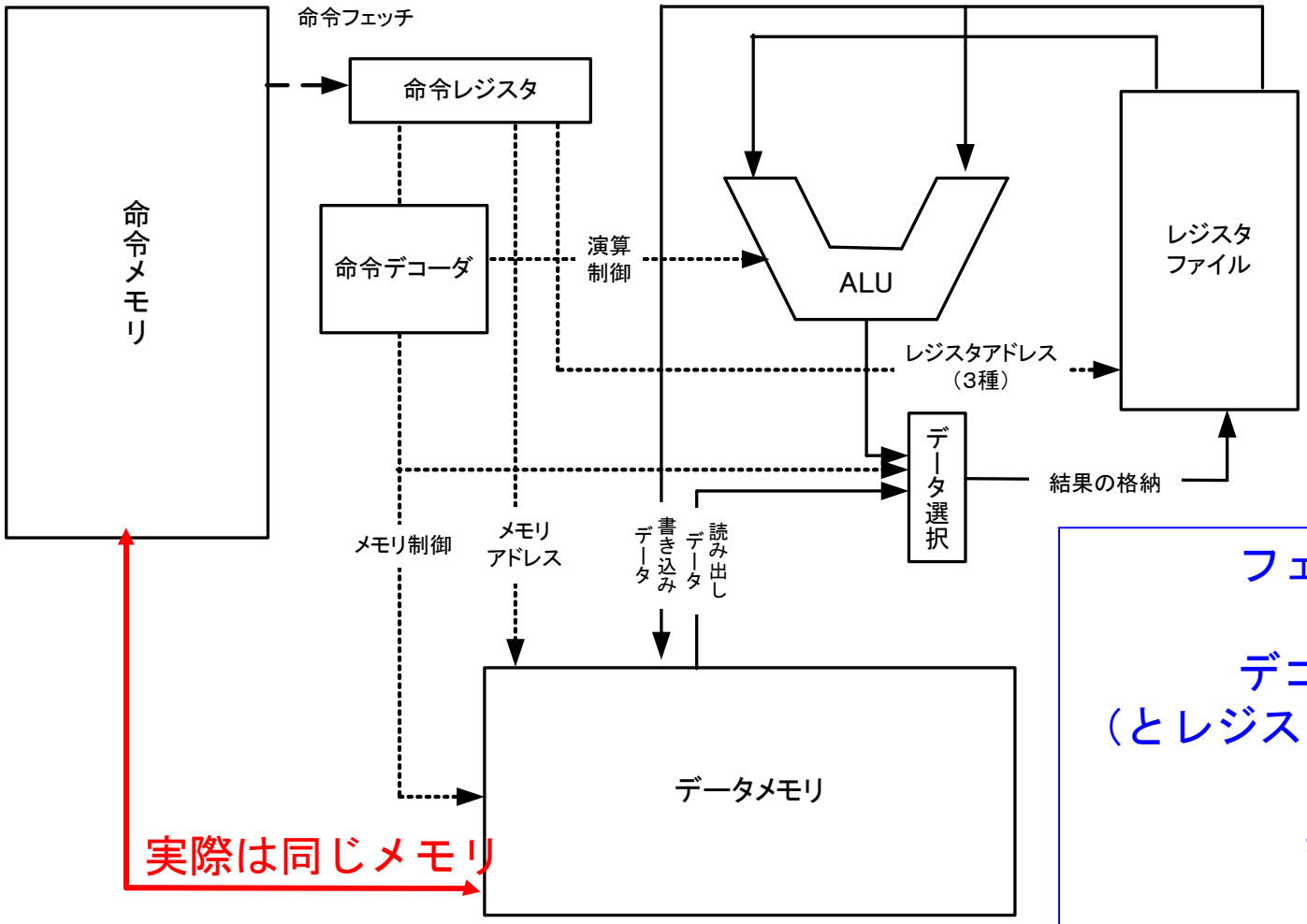
(2) メモリ操作命令



次の命令番地 ←「アドレス」

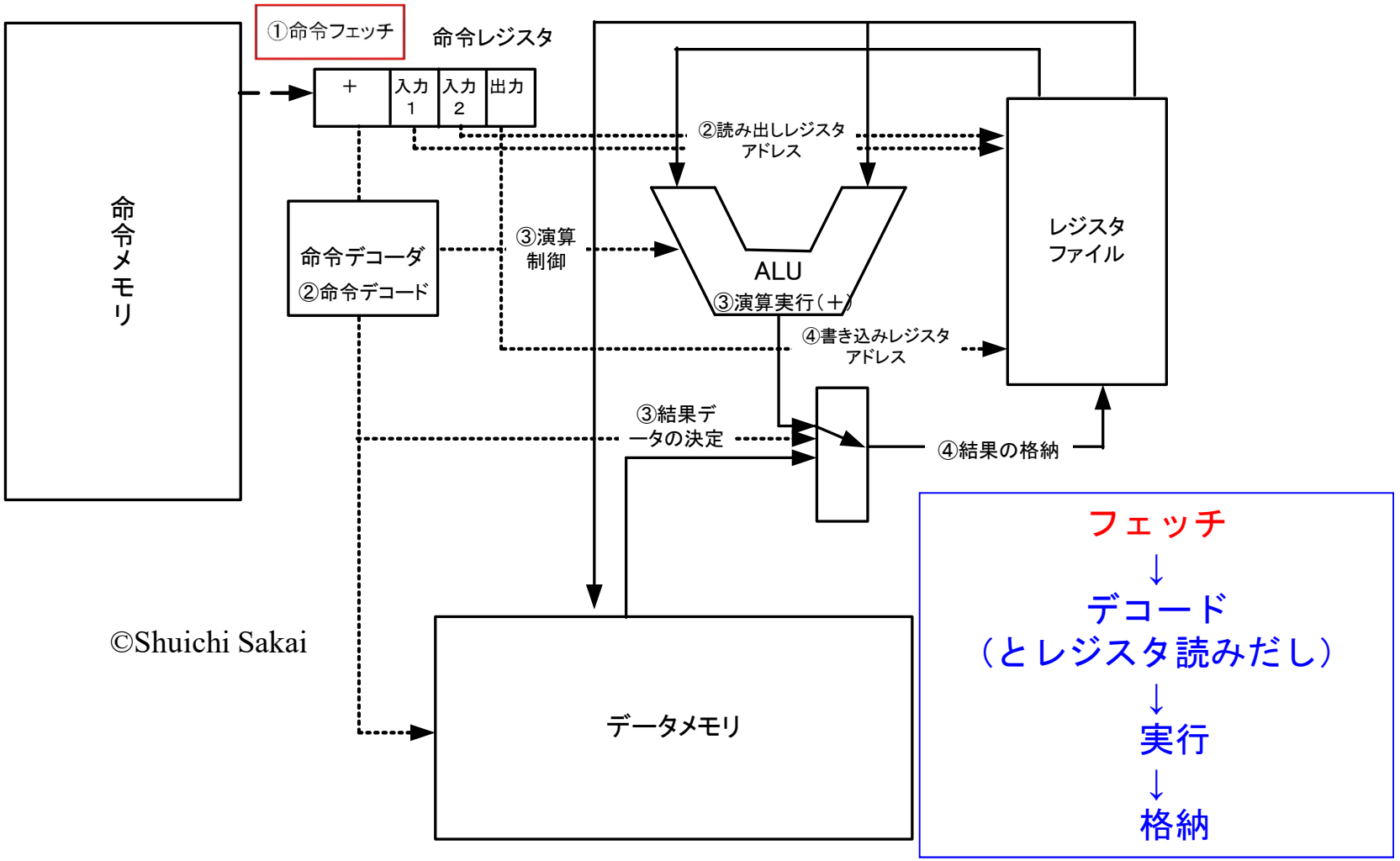
(3) 分岐命令

命令実行の基本形

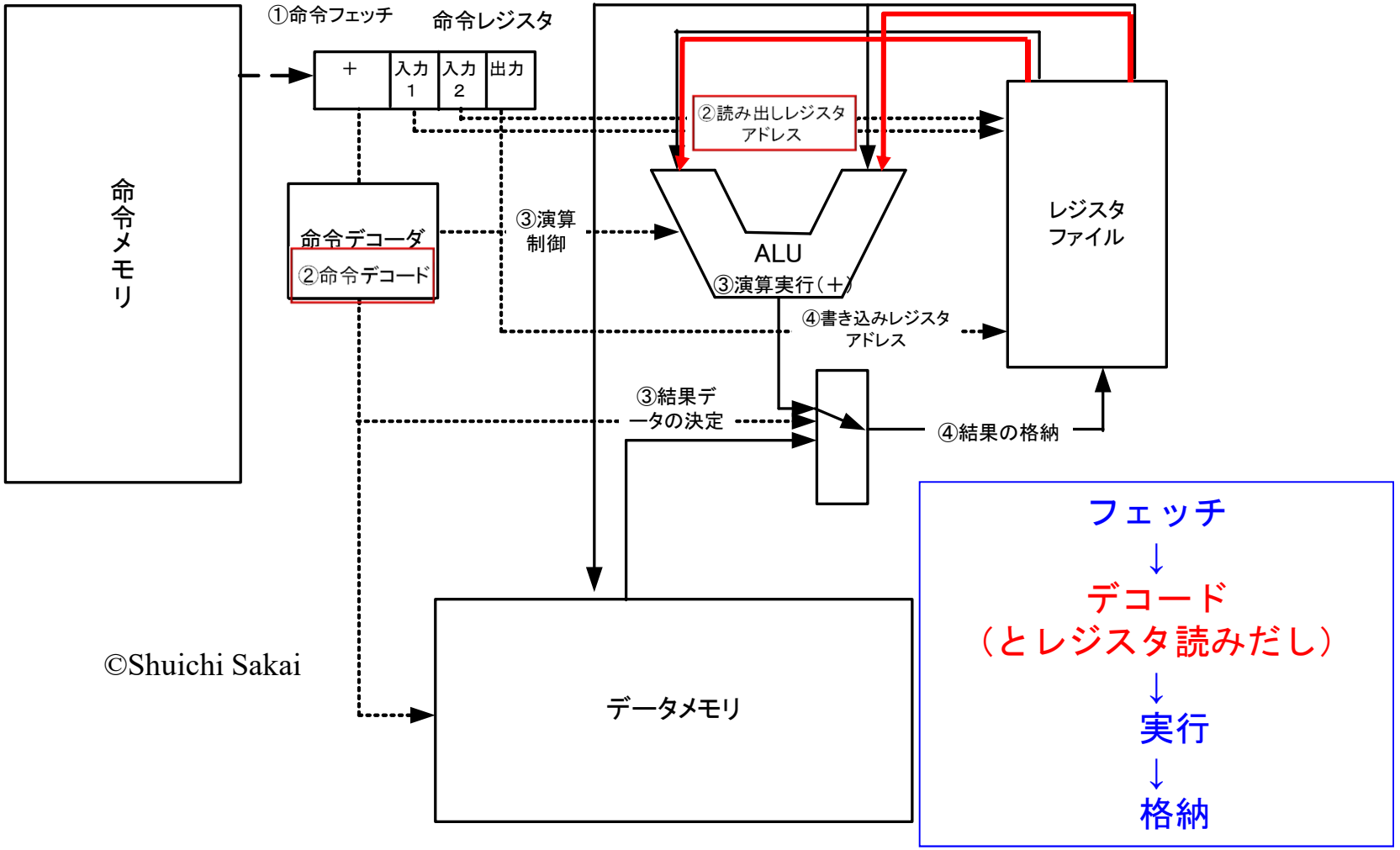


フェッチ
↓
デコード
（とレジスタ読みだし）
↓
実行
↓
格納

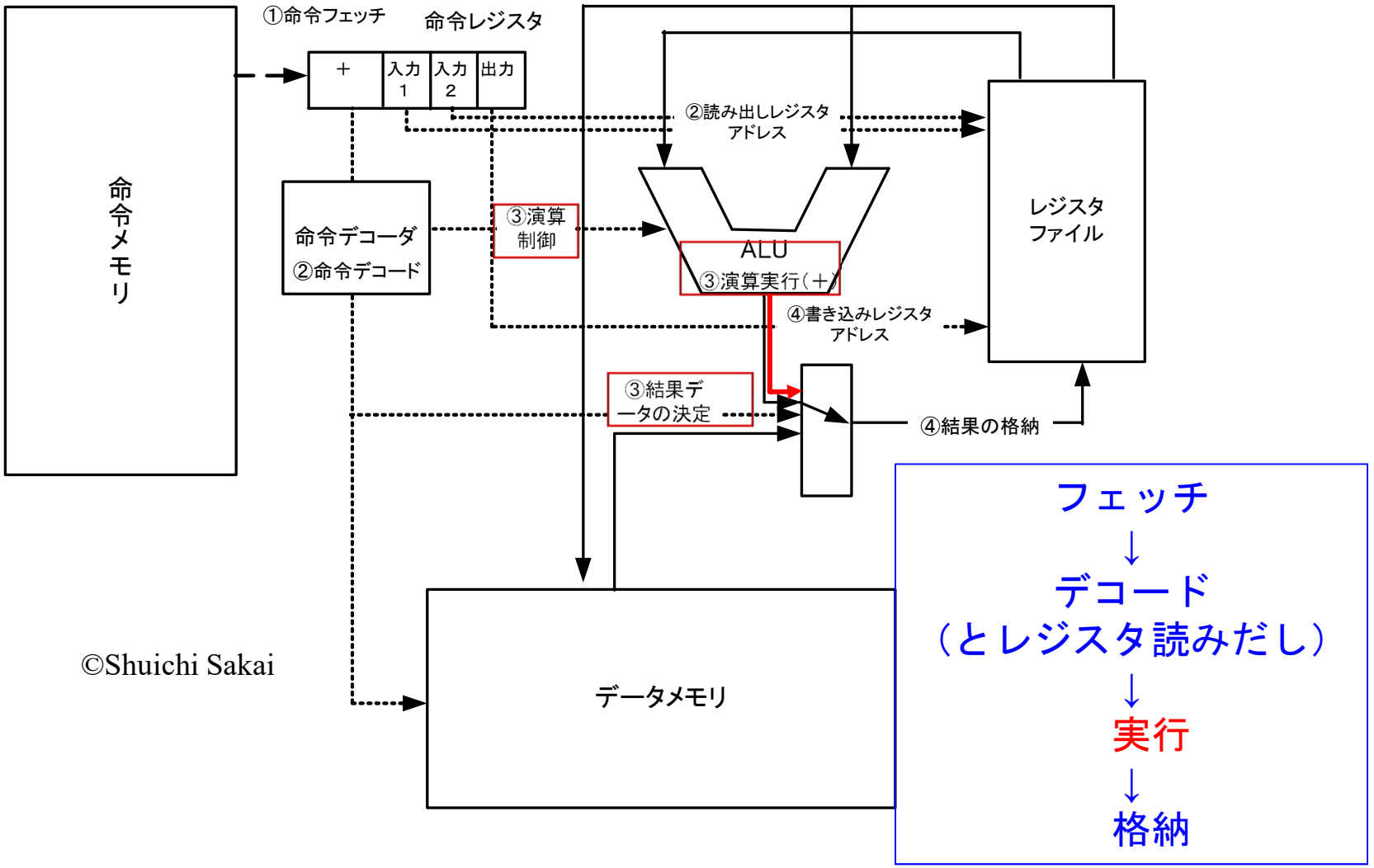
算術論理演算命令の実行サイクル



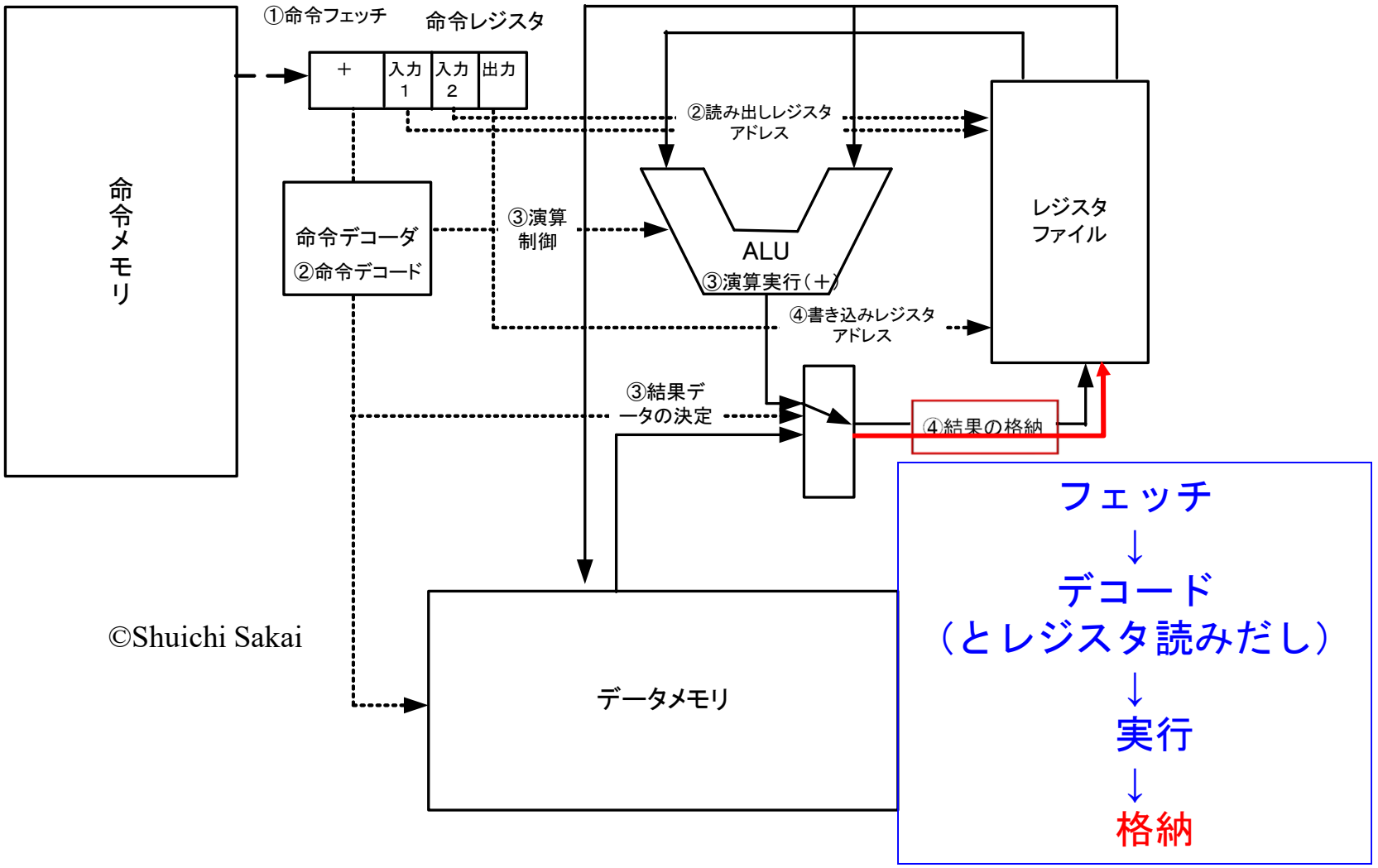
算術論理演算命令の実行サイクル



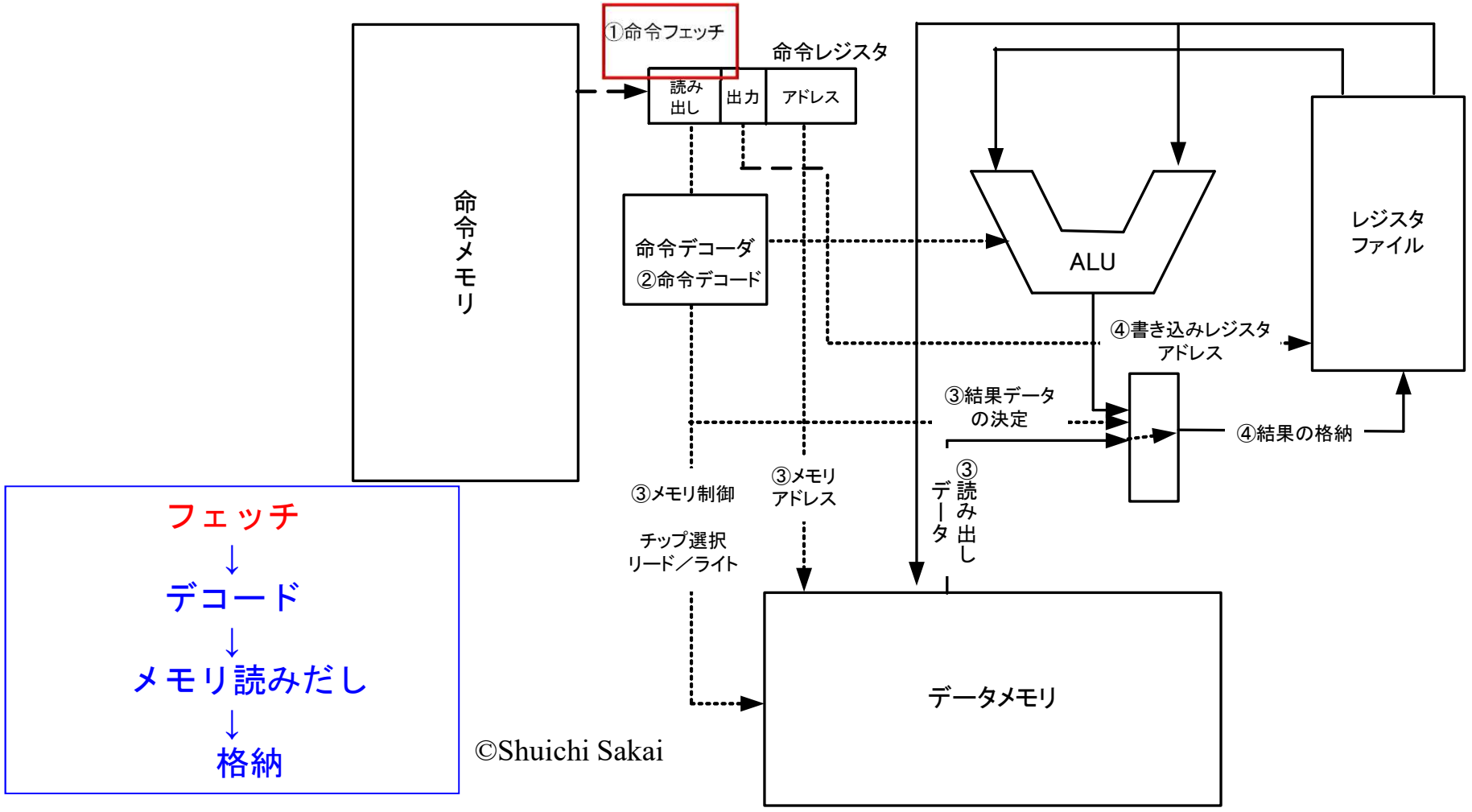
算術論理演算命令の実行サイクル



算術論理演算命令の実行サイクル

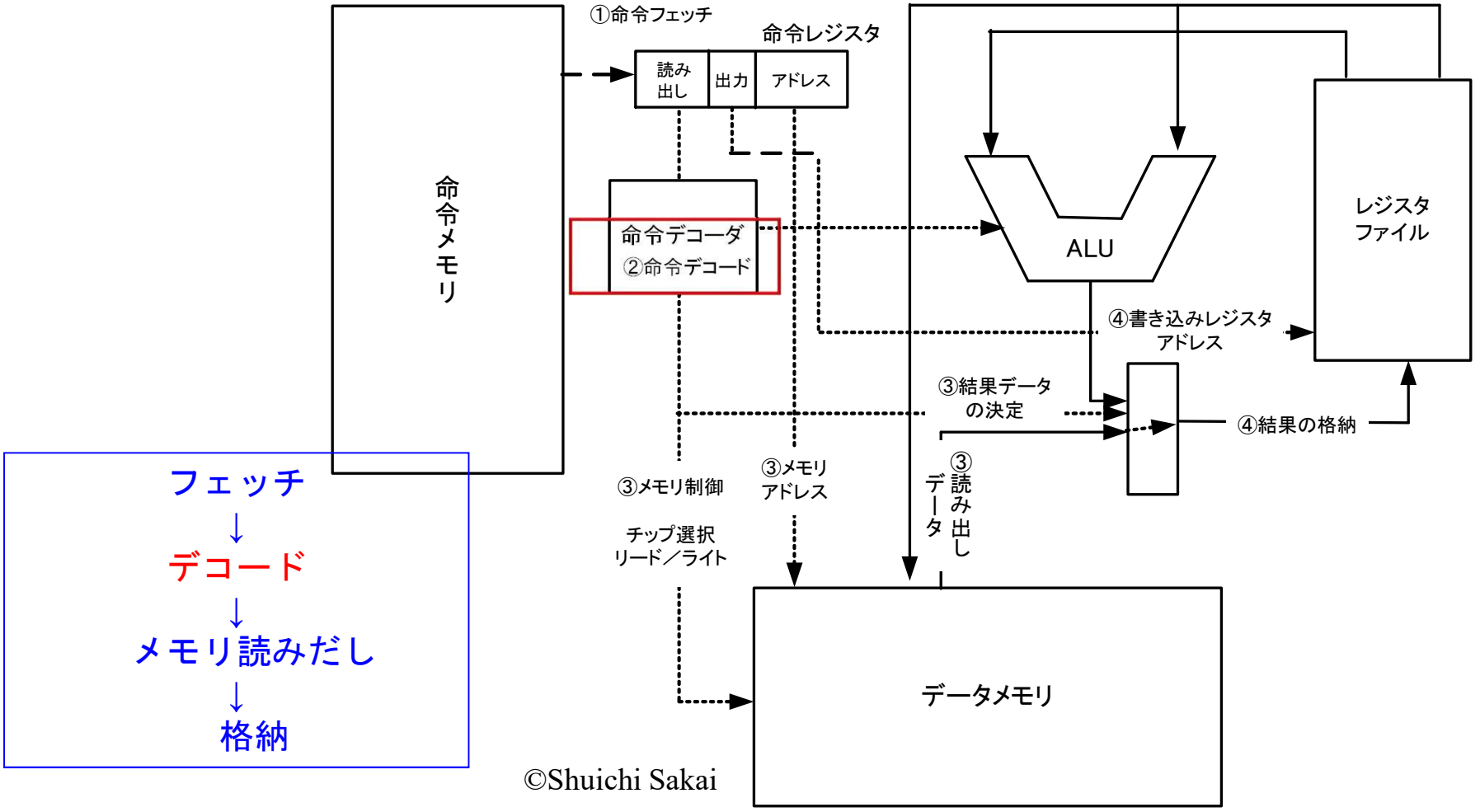


メモリ操作命令の実行サイクル
（読み出しの場合）

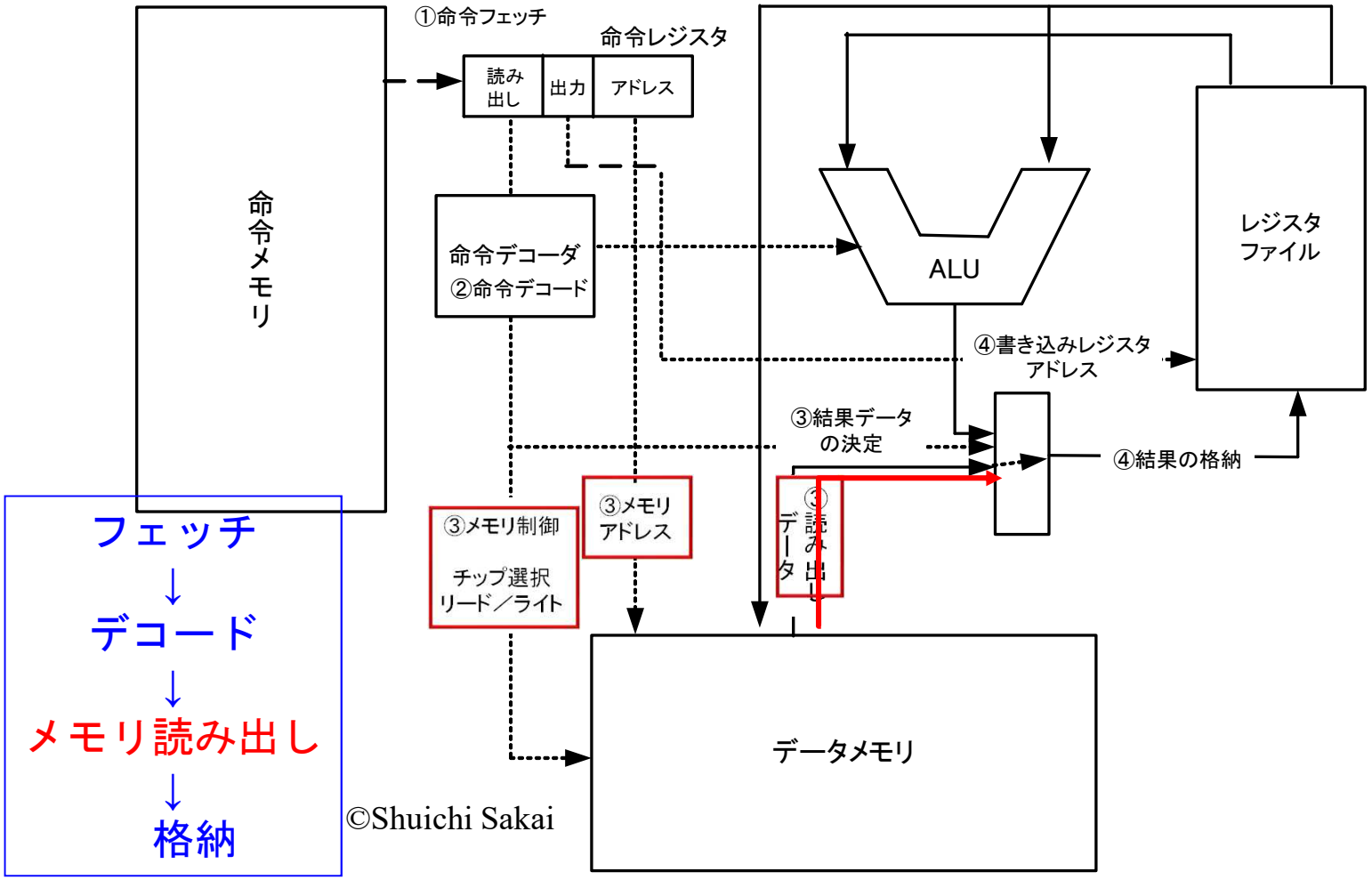


フェッチ
↓
デコード
↓
メモリ読みだし
↓
格納

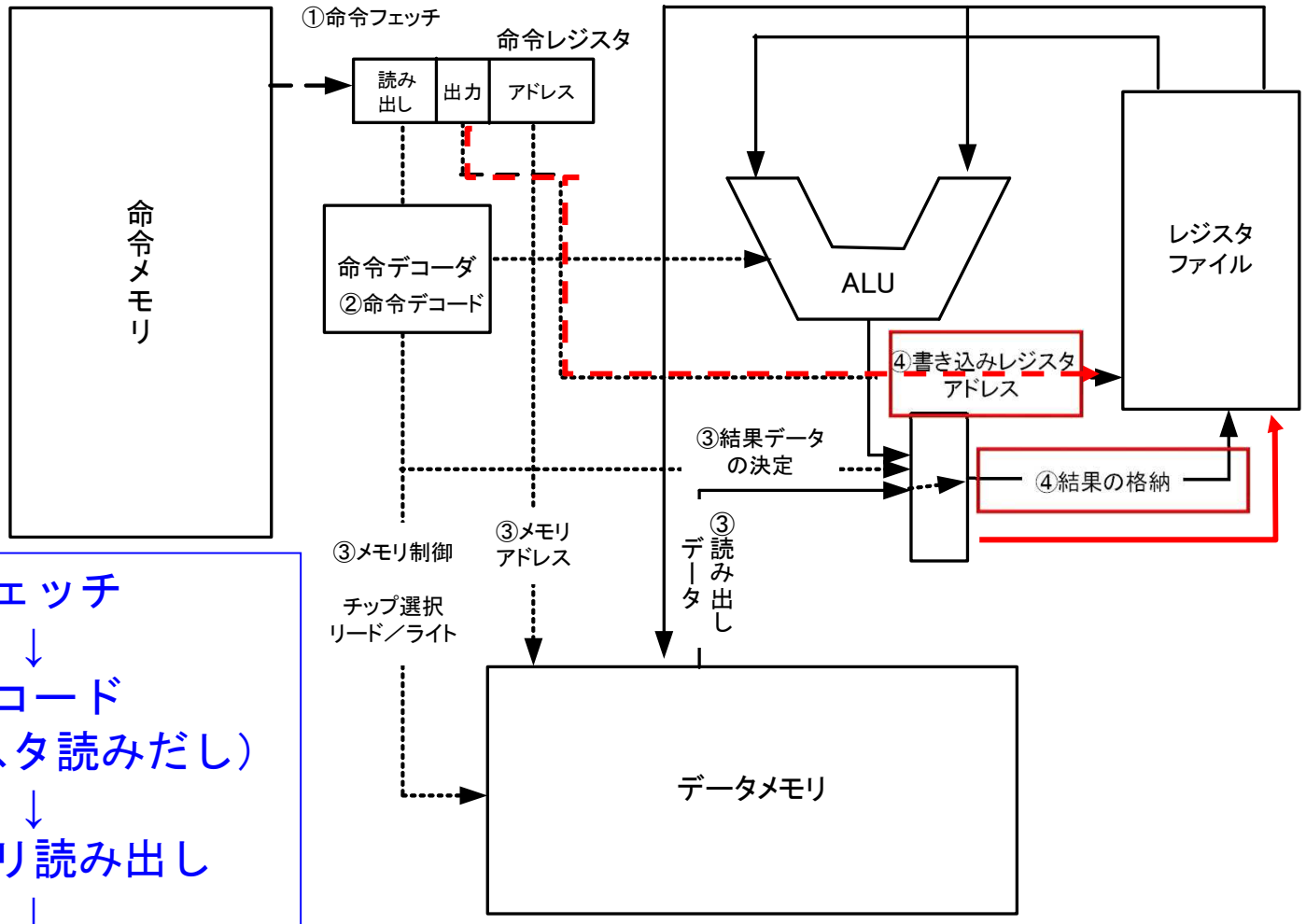
メモリ操作命令の実行サイクル
（読み出しの場合）



メモリ操作命令の実行サイクル
（読み出しの場合）

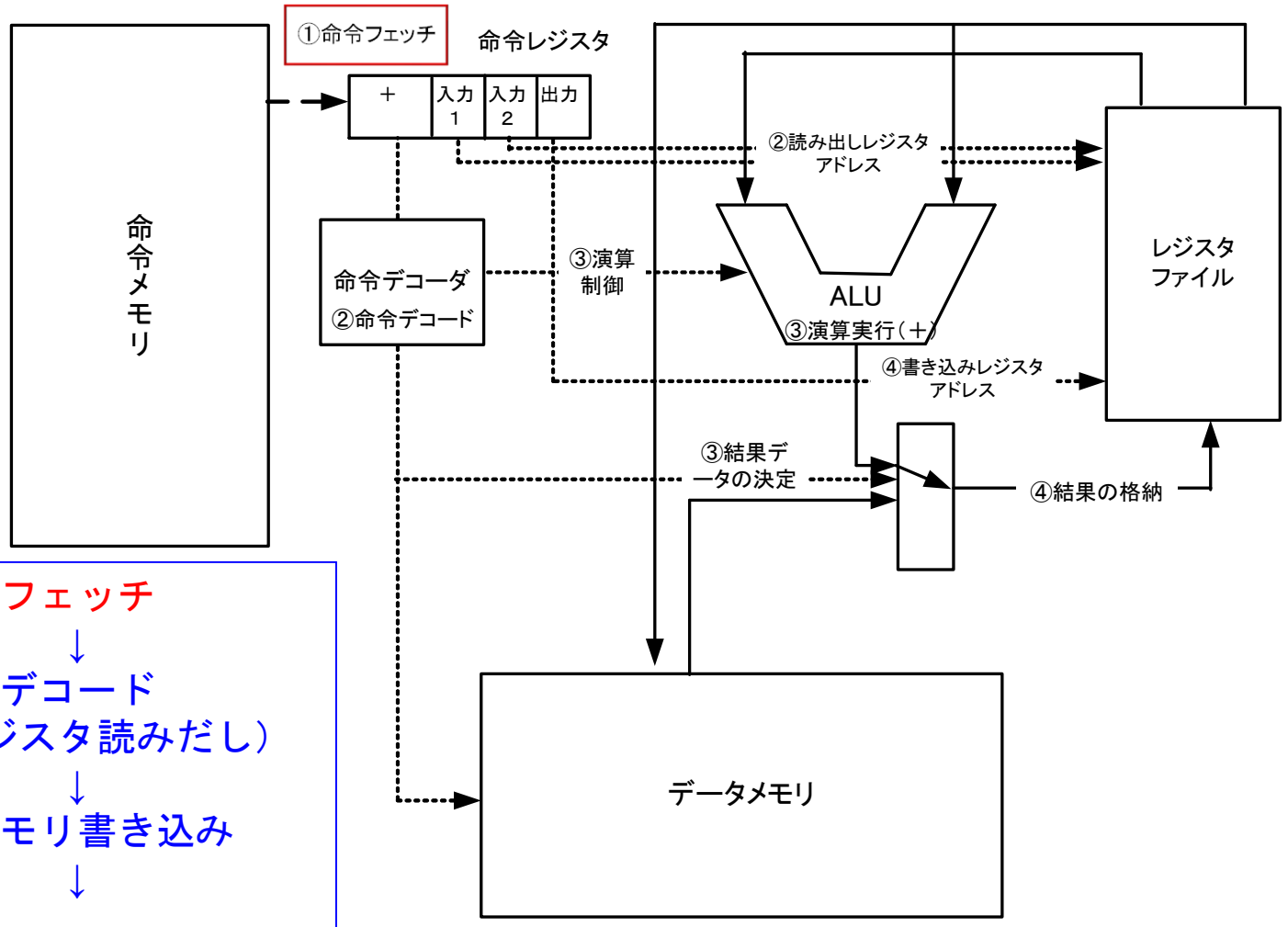


メモリ操作命令の実行サイクル
（読み出しの場合）



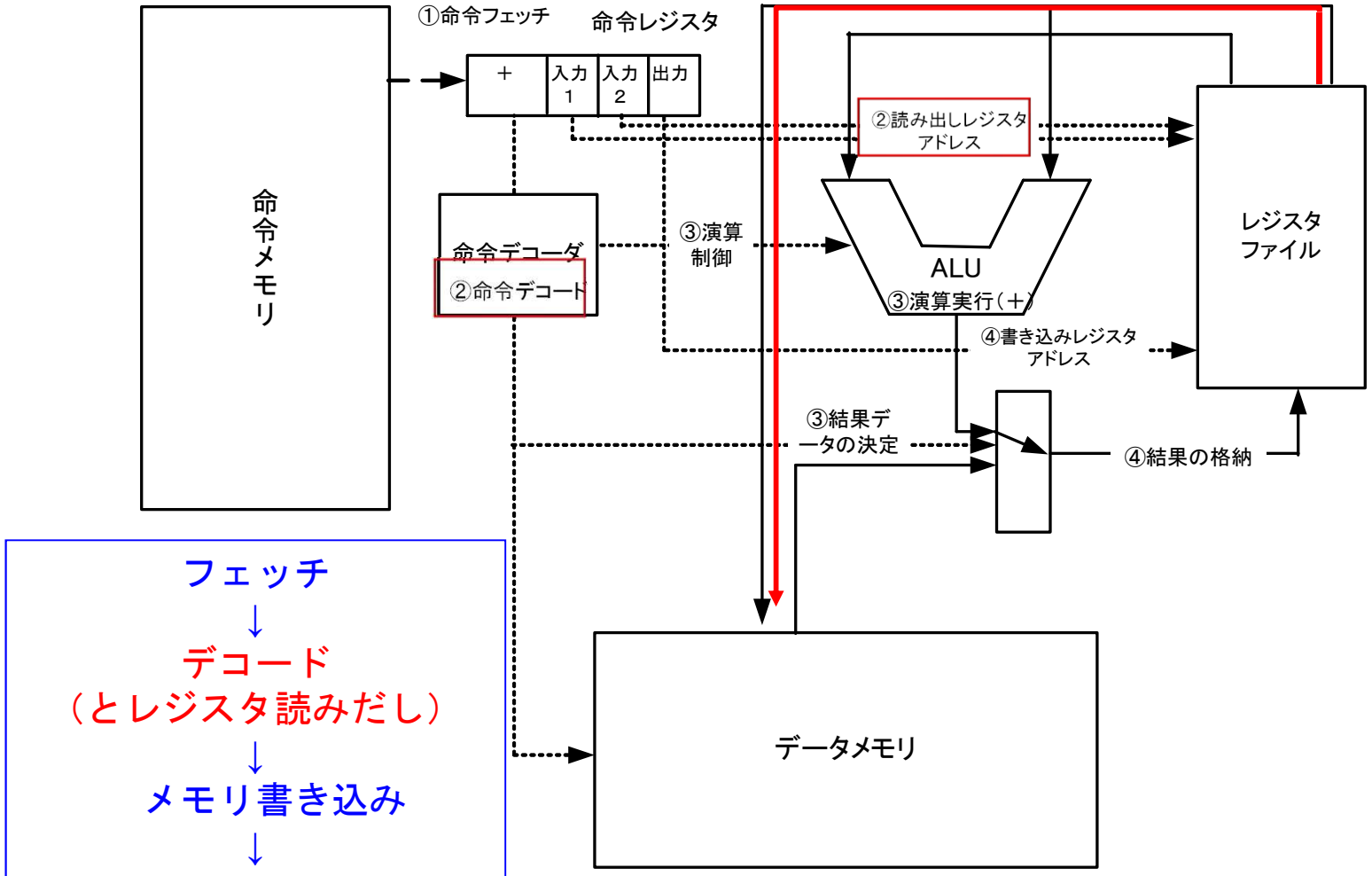
フェッチ
↓
デコード
（とレジスタ読みだし）
↓
メモリ読み出し
↓
格納

メモリ実行命令の実行サイクル（書き込み）



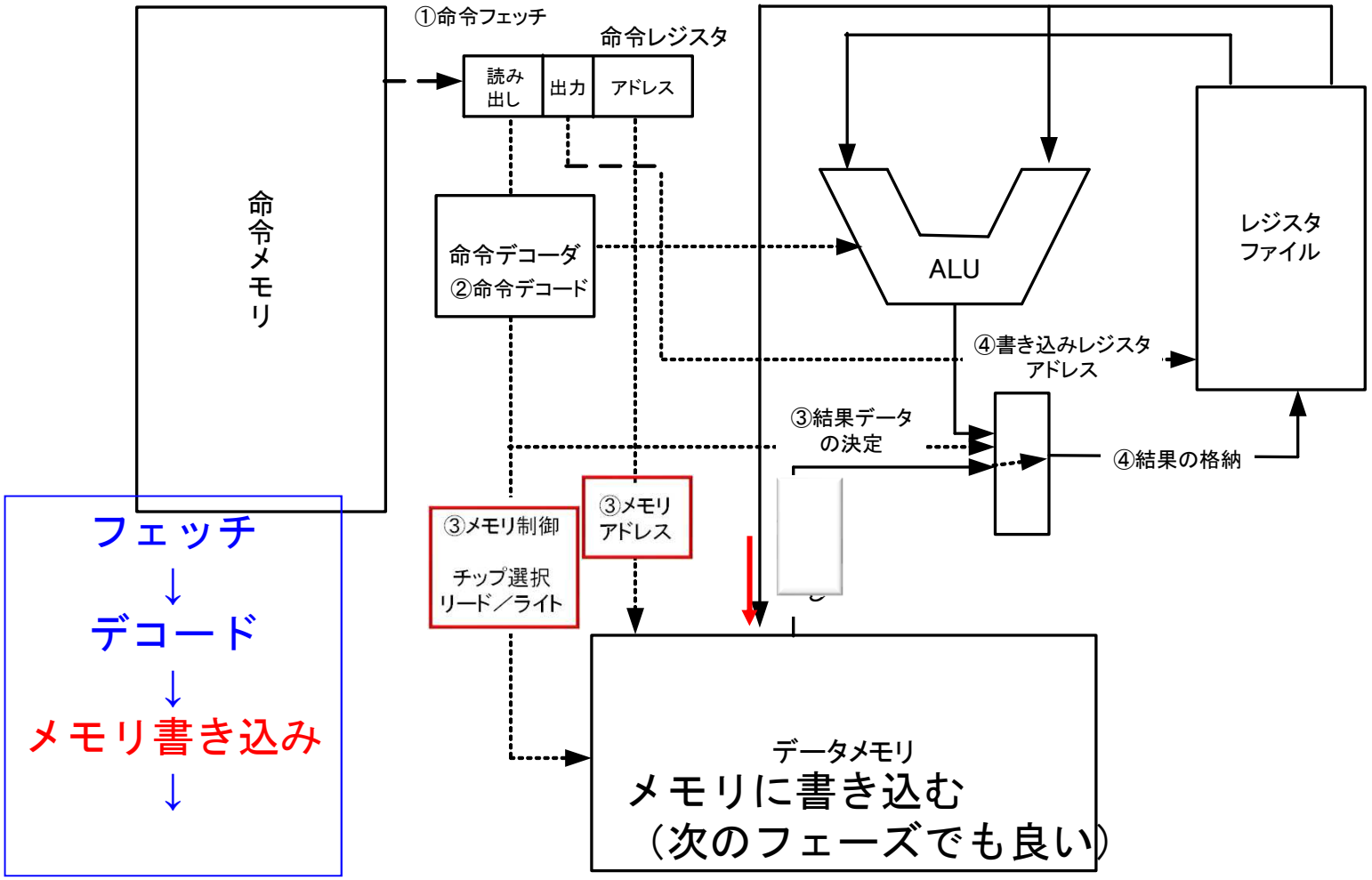
フェッチ
↓
デコード
（とレジスタ読みだし）
↓
メモリ書き込み
↓

メモリ実行命令の実行サイクル（書き込み）



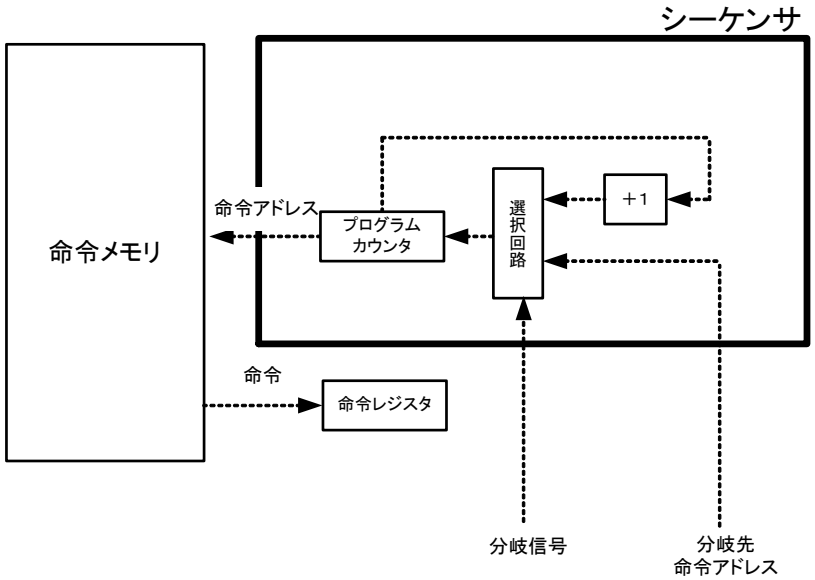
フェッチ
↓
デコード
（とレジスタ読みだし）
↓
メモリ書き込み
↓

メモリ操作命令の実行サイクル
（書き込み場合）



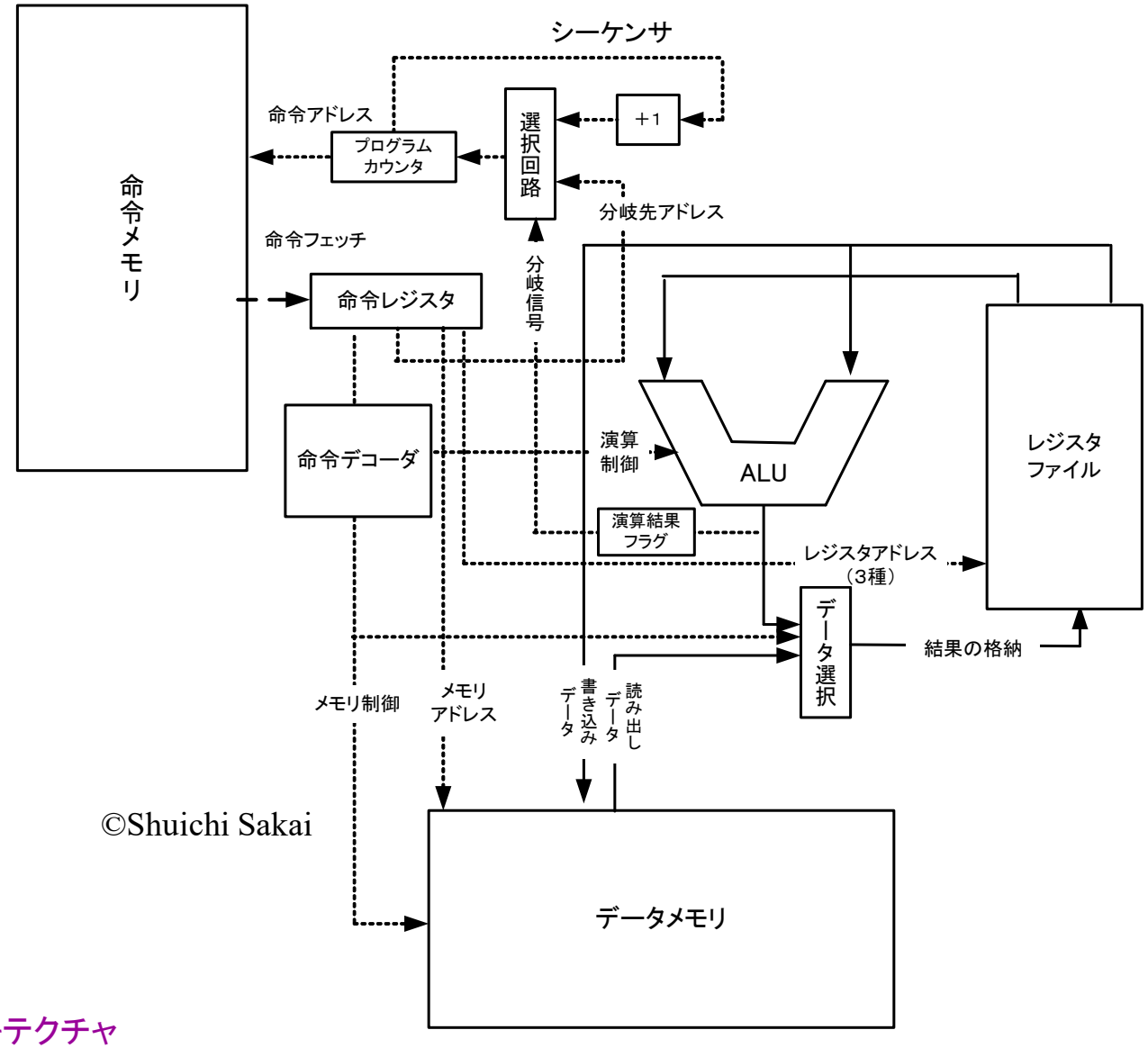
シーケンサ

シーケンサ　＝　次にどの命令を実行するのかを決める機構
　　＝　命令アドレス生成回路

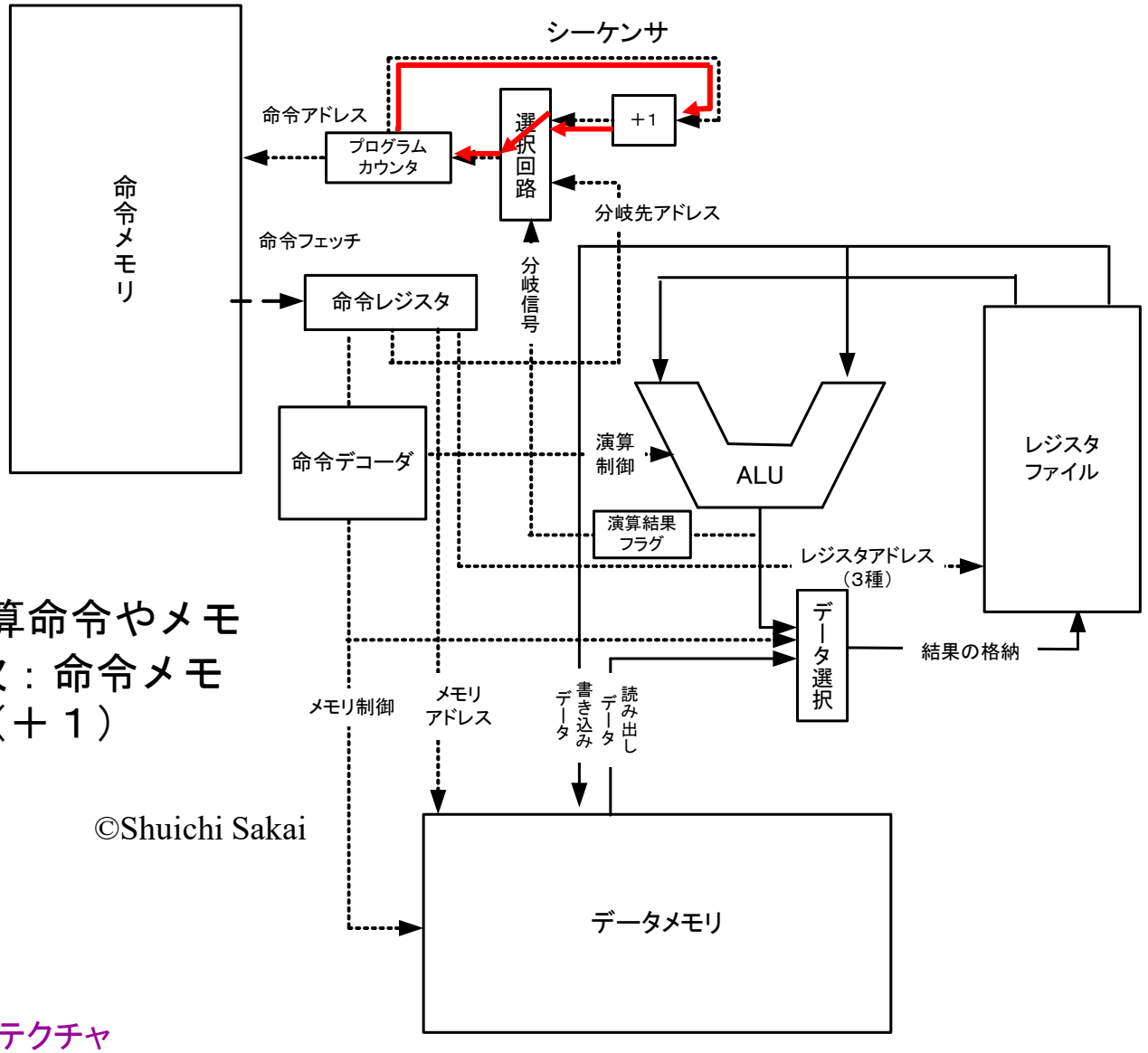


(1) 算術論理演算命令やメモリ操作命令の次：命令メモリの次の番地（＋１）
(2) 無条件分岐命令（ジャンプ命令）：ジャンプ先の番地命令（命令やレジスタから生成）
(3) 条件分岐命令（ブランチ命令）：　判定がYESなら分岐先の命令番地、NOなら「＋１」

コンピュータ中枢部の構成

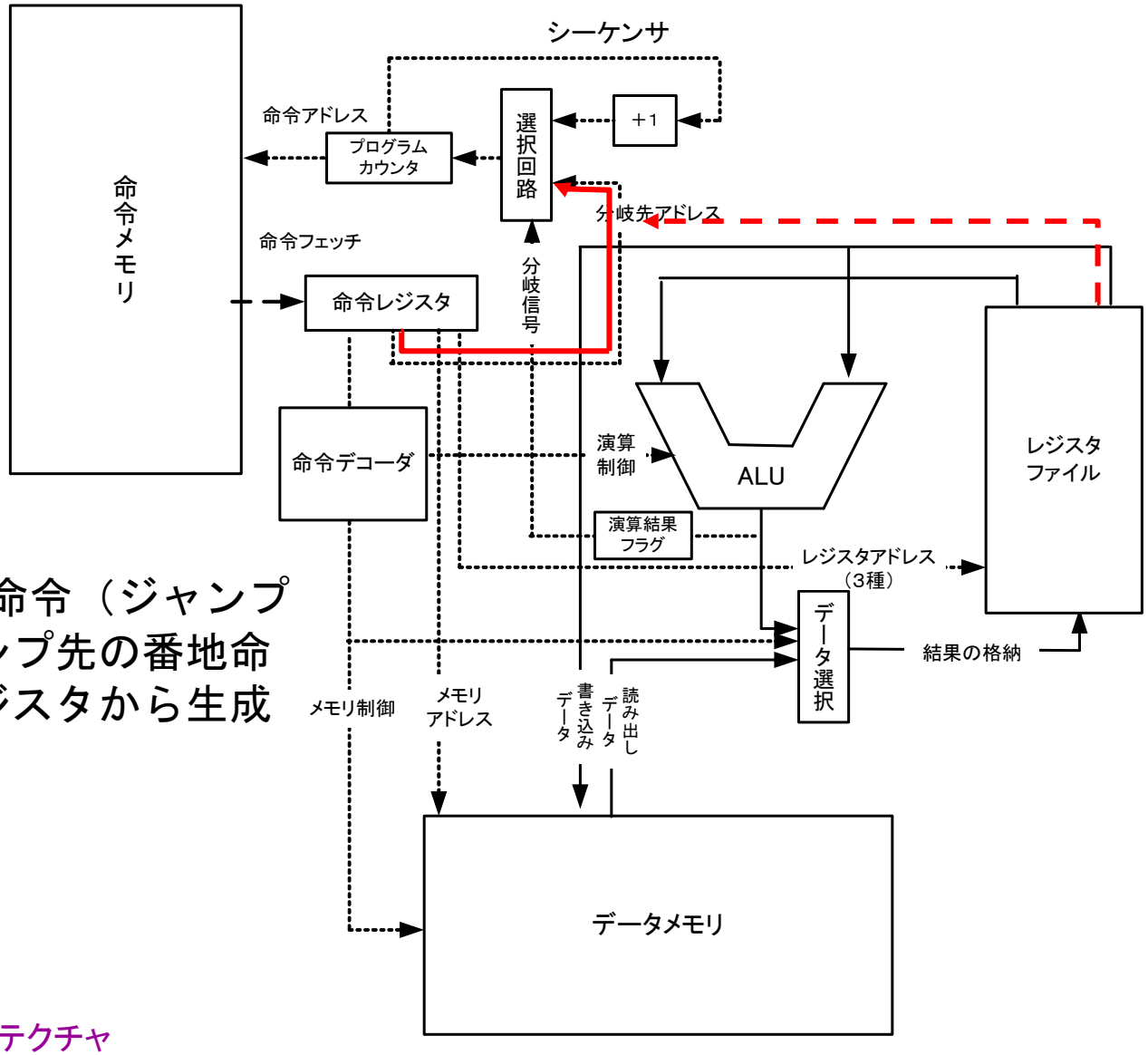


シーケンサの動き



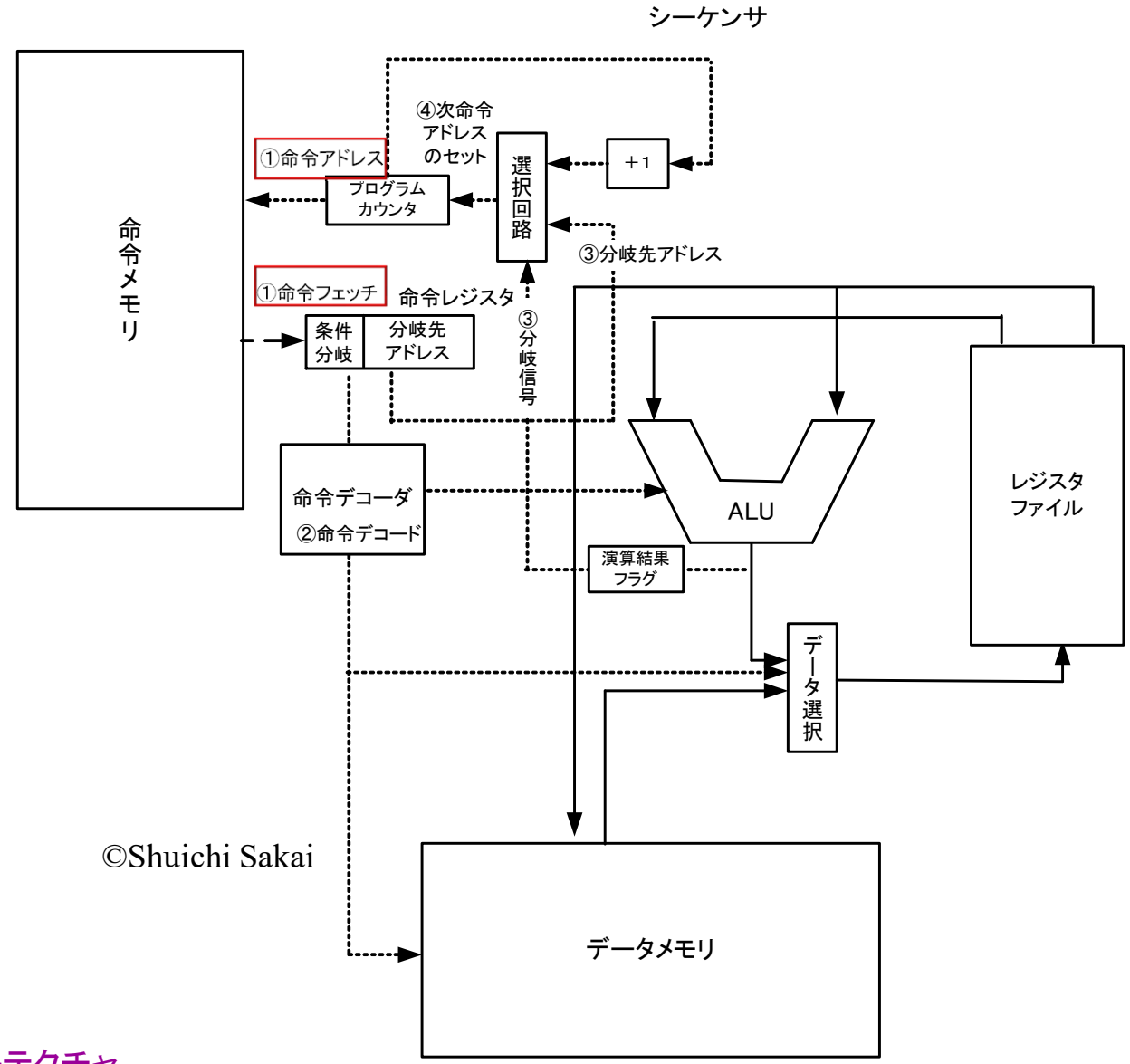
(1) 算術論理演算命令やメモリ操作命令の次：命令メモリの次の番地（＋１）

シーケンサの動き



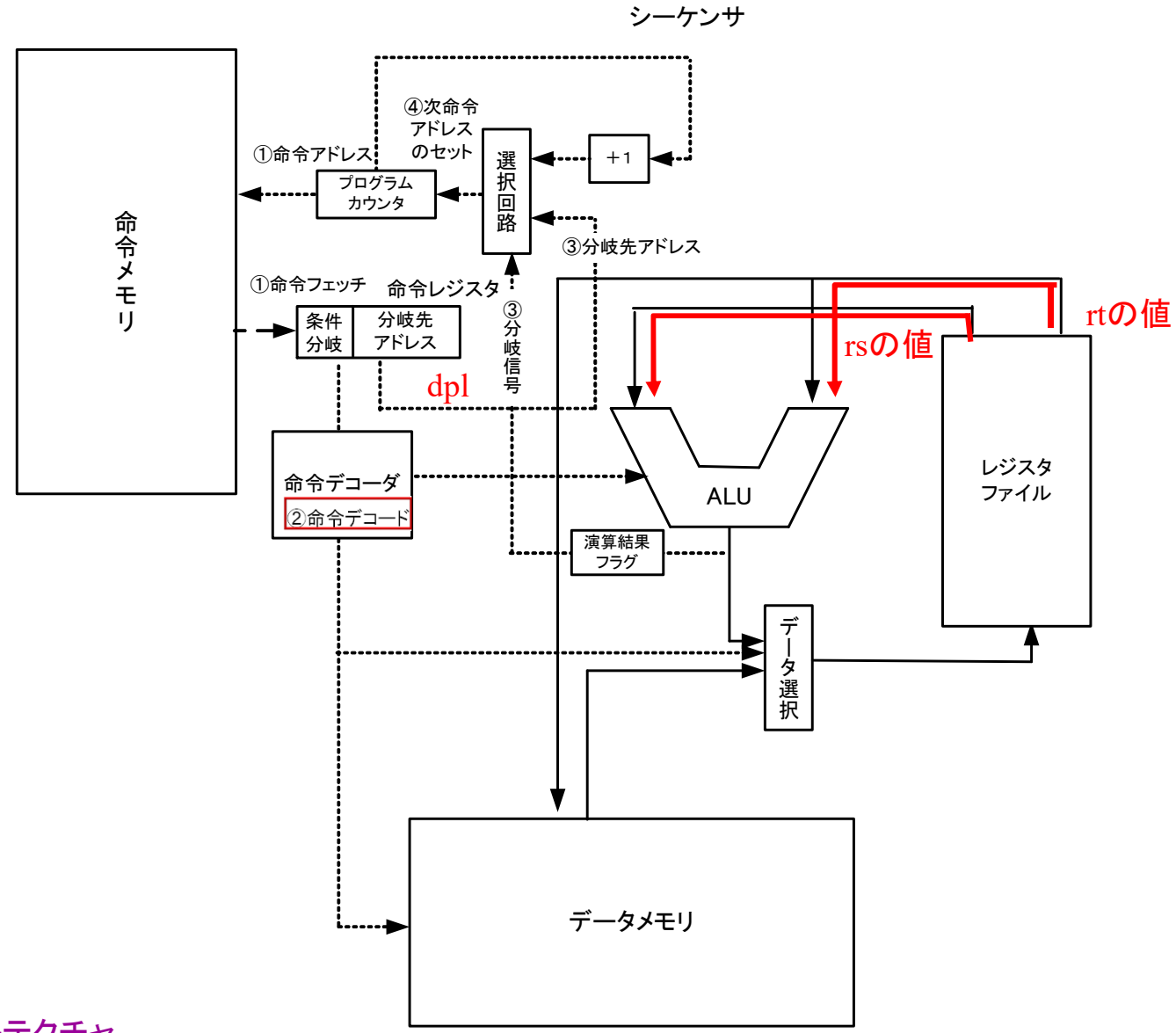
(2) 無条件分岐命令（ジャンプ命令）：ジャンプ先の番地命令（命令やレジスタから生成）

条件分岐命令の実行サイクル



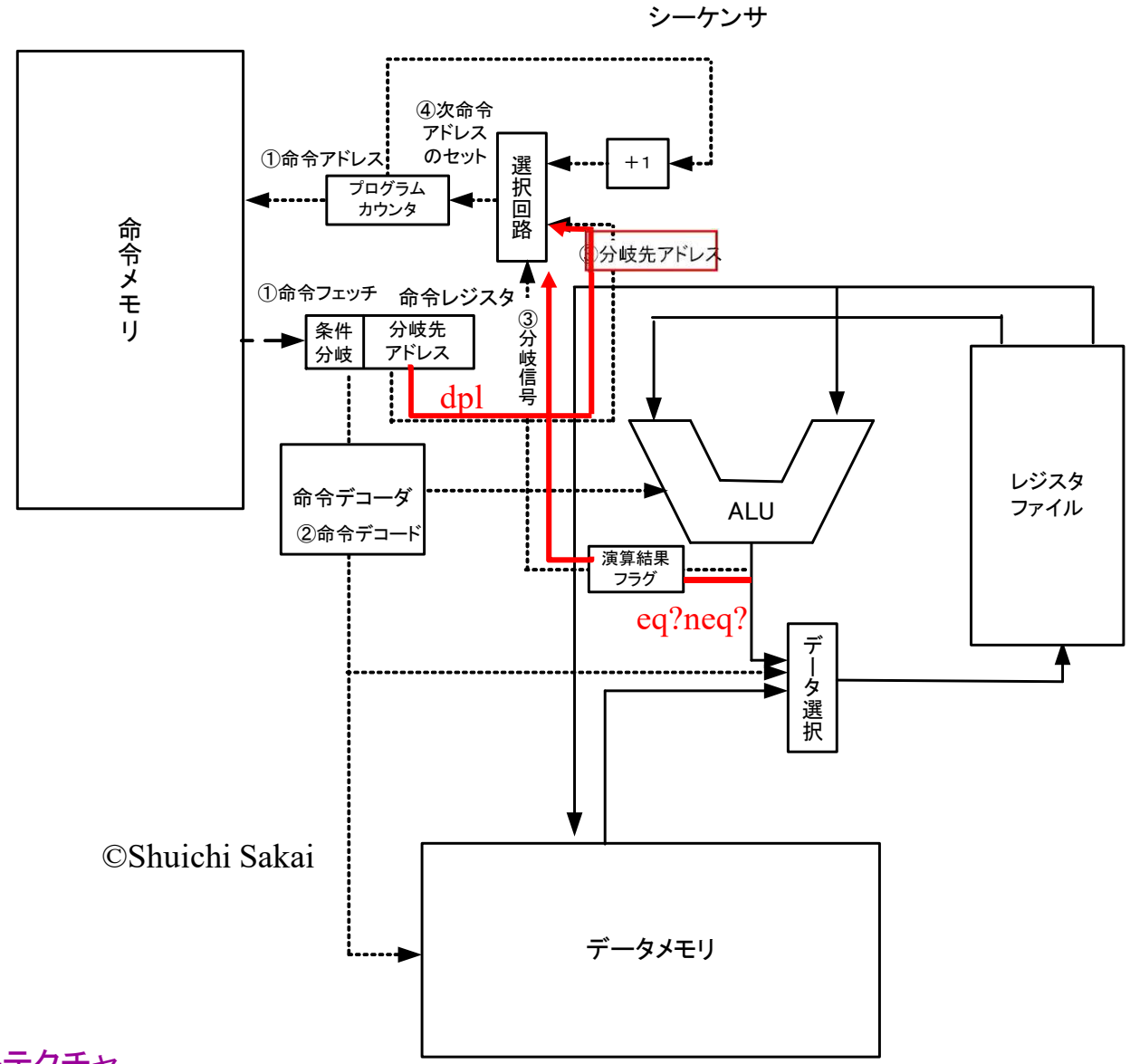
bne rs,rt,dpl

条件分岐命令の実行サイクル



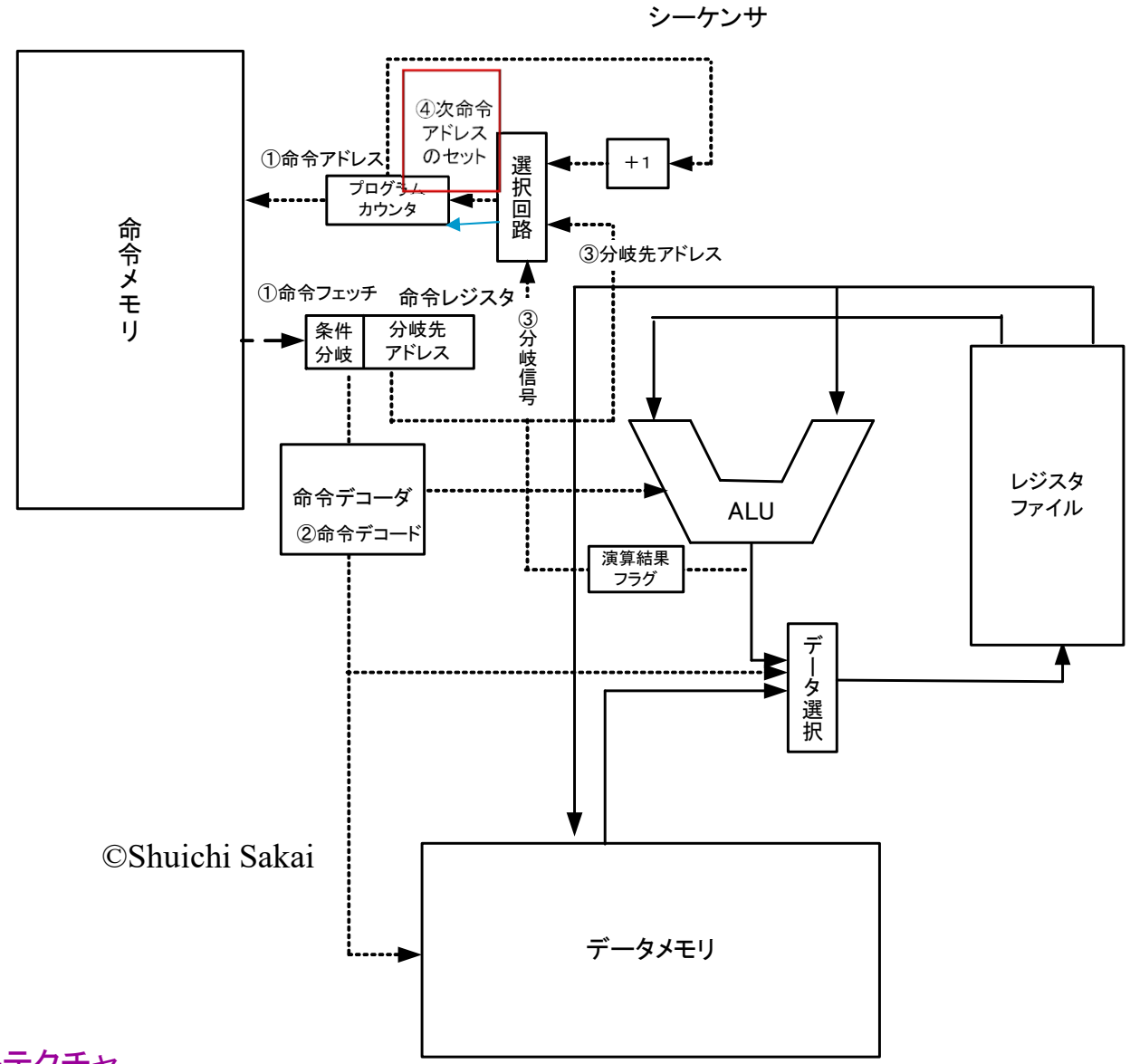
bne rs,rt,dpl

条件分岐命令の実行サイクル

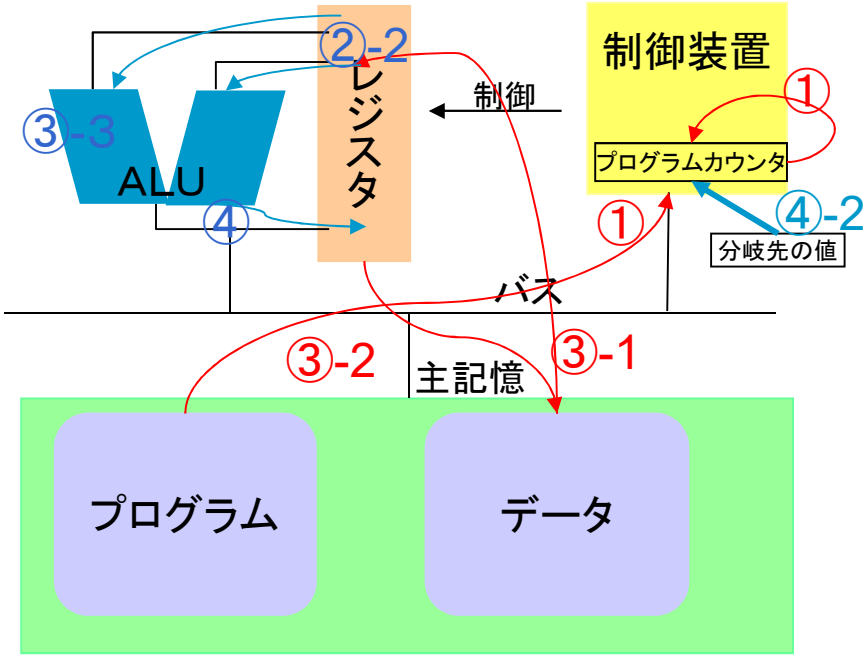


bne rs,rt,dpl

条件分岐命令の実行サイクル



ノイマン型計算機の仕組み



- **ノイマン型計算機**は、プログラム内蔵方式のコンピュータ・アーキテクチャ
  - 主記憶装置（メモリ）上に命令とデータを格納
- 演算論理装置（ALU）、レジスタ制御装置、メモリ、と、これらを接続するバスから構成
- 以下のようなステップを繰り返し行うことで計算を実行

①プログラムカウンタのさすアドレスから次の命令を読み込む（かつプログラムカウンタを命令長だけ増加）(Fetch)

②制御装置で命令を解釈する。演算命令の場合はレジスタから値を読む（②−2）(Decode)

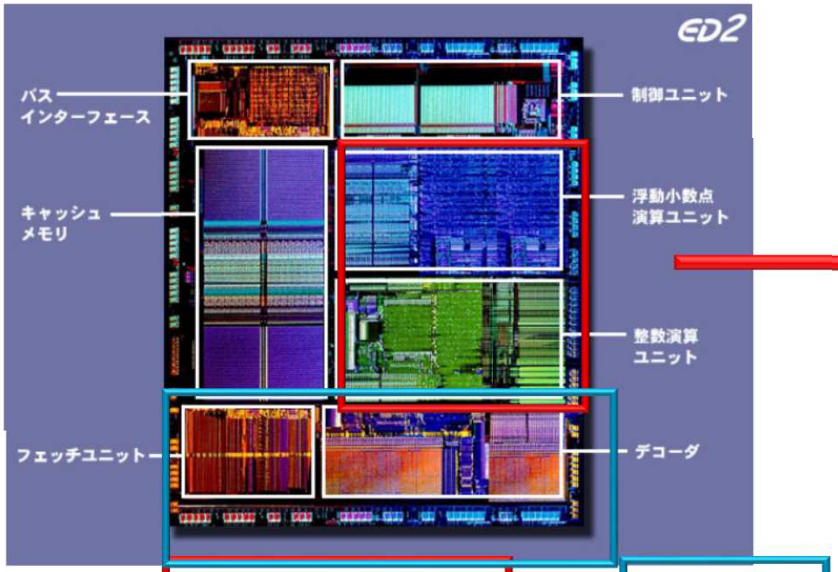
③命令に従いデータを主記憶から読み出す（③−1）/主記憶に書き込む。(③−2)
レジスタのデータを用いて演算を行う。(③−③) (Execute)

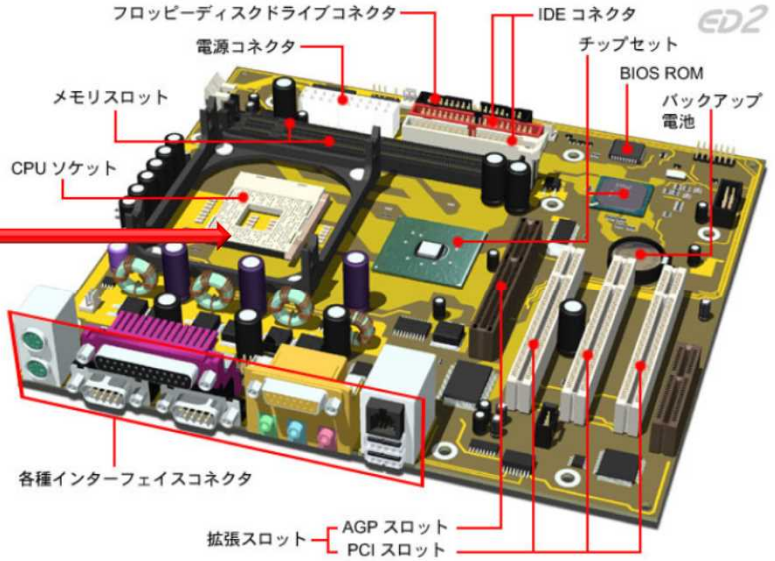
④演算結果をレジスタに書き込む。(④−1)分岐命令の場合はプログラムカウンタの値を新しい値に設定する（④−2）①に戻る (Write)

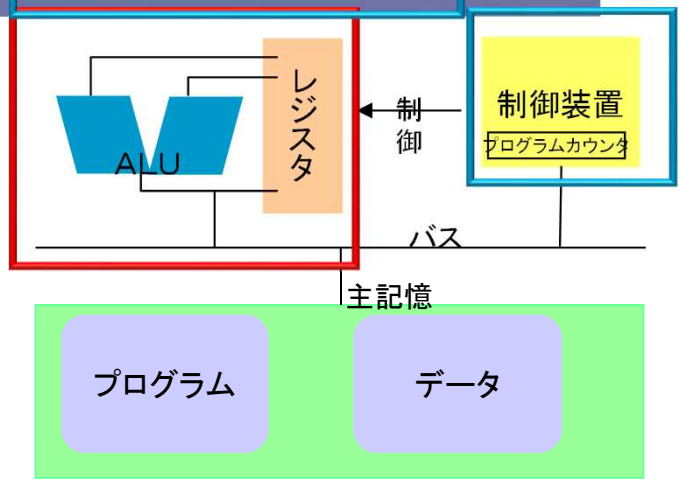
【特徴】（メモ）

- 制御が非常に柔軟
- ひとつの命令で主記憶の1個のデータの状態しか変更できない
- 逐一プログラムとデータを主記憶から取ってくるため、主記憶の性能がネック
  → Von Neumann ボトルネック

CPU(中央処理装置)

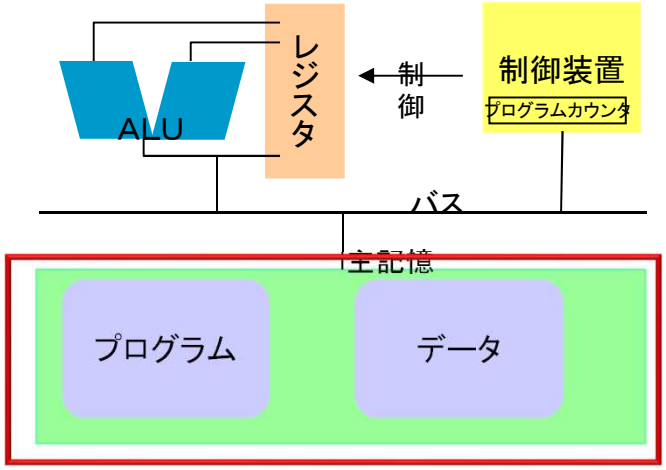


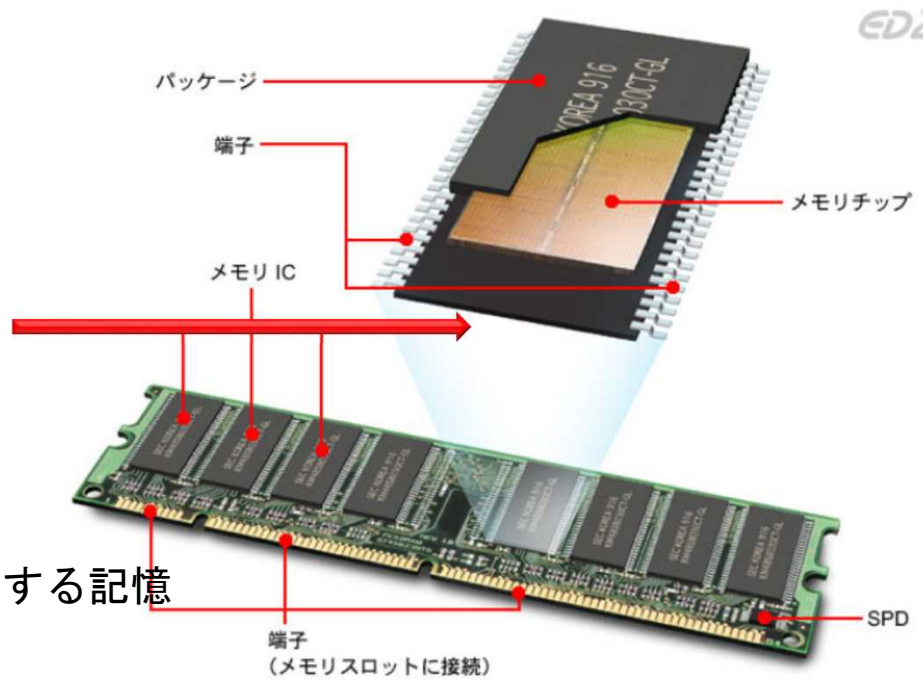




出典：http://www.sugilab.net/jk/joho-kiki/index.html

主記憶(メモリ)





主記憶は名前の示すとおり、
プログラムやデータなどを大量に保持する記憶装置。

どのデータを参照するかを示すためにアドレス（住所）でどのデータを欲しいかを指定する。

通常はアドレスはバイト（8ビット）単位でつけられる。

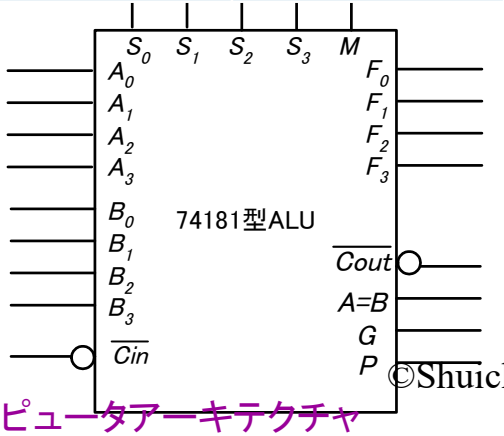
アドレス	そのアドレスに保持されるデータ
0	0x00000001
4	0x12345678
8	0xabcdef01

出典：http://www.sugilab.net/jk/joho-kiki/index.html

演習問題

下記の命令コードを有する
演算器を74181型ALUで構成する
ときS3,S2,S1,S0,M、Cin'の論理式を
記述せよ。

演算種	コード (Op2,Op1,OP0)
加算	000
減算	001
AND	010
OR	011
NOT A	100



制御信号 $S_3\ S_2\ S_1\ S_0$	M=1: 論理演算	M=0: 算術演算 $\overline{Cin}$=0	M=0: 算術演算 $\overline{Cin}$=1
0 0 0 0	$F=\overline{A}$	$F=A$	$F=A$ **PLUS 1**
0 0 0 1	$F=\overline{A+B}$	$F=A+B$	$F=(A+B)$ PLUS 1
0 0 1 0	$F=\overline{A}\cdot B$	$F=A+\overline{B}$	$F=(A+\overline{B})$ PLUS 1
0 0 1 1	$F=0$	$F=$ 1111	$F=$ZERO
0 1 0 0	$F=\overline{A\cdot B}$	$F=A$ PLUS $A\cdot\overline{B}$	$F=A$ PLUS $A\cdot\overline{B}$ PLUS 1
0 1 0 1	$F=\overline{B}$	$F=(A+B)$ PLUS $A\cdot\overline{B}$	$F=(A+B)$ PLUS $A\cdot\overline{B}$ PLUS 1
0 1 1 0	$F=A\oplus B$	$F=A$ **MINUS** B **MINUS 1**	$F=A$ **MINUS** B
0 1 1 1	$F=A\cdot\overline{B}$	$F=A\cdot\overline{B}$ MINUS 1	$F=A\cdot\overline{B}$
1 0 0 0	$F=\overline{A}+B$	$F=A$ PLUS $A\cdot B$	$F=A$ PLUS $A\cdot B$ PLUS 1
1 0 0 1	$F=\overline{A\oplus B}$	$F=A$ **PLUS** B	$F=A$ **PLUS** B **PLUS 1**
1 0 1 0	$F=B$	$F=(A+\overline{B})$ PLUS AB	$F=(A+\overline{B})$ PLUS $A\cdot B$ PLUS 1
1 0 1 1	$F=A\cdot B$	$F=A\cdot B$ MINUS 1	$F=A\cdot B$
1 1 0 0	$F=1$	$F=A$ PLUS A	$F=A$ PLUS A PLUS 1
1 1 0 1	$F=A+\overline{B}$	$F=(A+B)$ PLUS A	$F=(A+B)$ PLUS A PLUS 1
1 1 1 0	$F=A+B$	$F=(A+\overline{B})$ PLUS A	$F=(A+\overline{B})$ PLUS A PLUS 1
1 1 1 1	$F=A$	$F=A$ **MINUS 1**	$F=A$

（b）動作：　重要なものは太字とした

図 1.10　74181 型 ALU（4 ビット）（前ページからの続き）

回答

Op2	Op1	Op0	S3	S2	S1	S0	M	Cin’
0	0	0	1	0	0	1	0	0
0	0	1	0	1	1	0	0	1
0	1	0	1	0	1	1	1	dc
0	1	1	1	1	1	0	1	dc
1	Dc	Dc	0	0	0	0	1	dc

S3 OP0

OP2,OP1	0	1
00	1	0
01	1	1
11	0	0
10	0	0

S3=OP2’*OP0’+ OP2’ * OP1

S2 OP0

OP2,OP1	0	1
00	0	1
01	0	1
11	0	0
10	0	0

S2=OP2’*OP0

S1 OP0

OP2,OP1	0	1
00	0	1
01	1	1
11	0	0
10	0	0

S1=OP2’*OP0+ OP2’*OP1

S0 OP0

OP2,OP1	0	1
00	1	0
01	1	0
11	0	0
10	0	0

S0=OP2’*OP0’

回答

Op2	Op1	Op0	S3	S2	S1	S0	M	Cin’
0	0	0	1	0	0	1	0	0
0	0	1	0	1	1	0	0	1
0	1	0	1	0	1	1	1	dc
0	1	1	1	1	1	0	1	dc
1	Dc	Dc	0	0	0	0	1	dc

M　OP0

OP2,OP1	0	1
00	0	0
01	1	1
11	1	1
10	1	1

M=OP2+OP1

Cin1’ OP0

OP2,OP1	0	1
00	0	1
01	dc	dc
11	dc	dc
10	dc	dc

Cin’=OP1