第１回

計算機の仕組みと機械語

佐藤

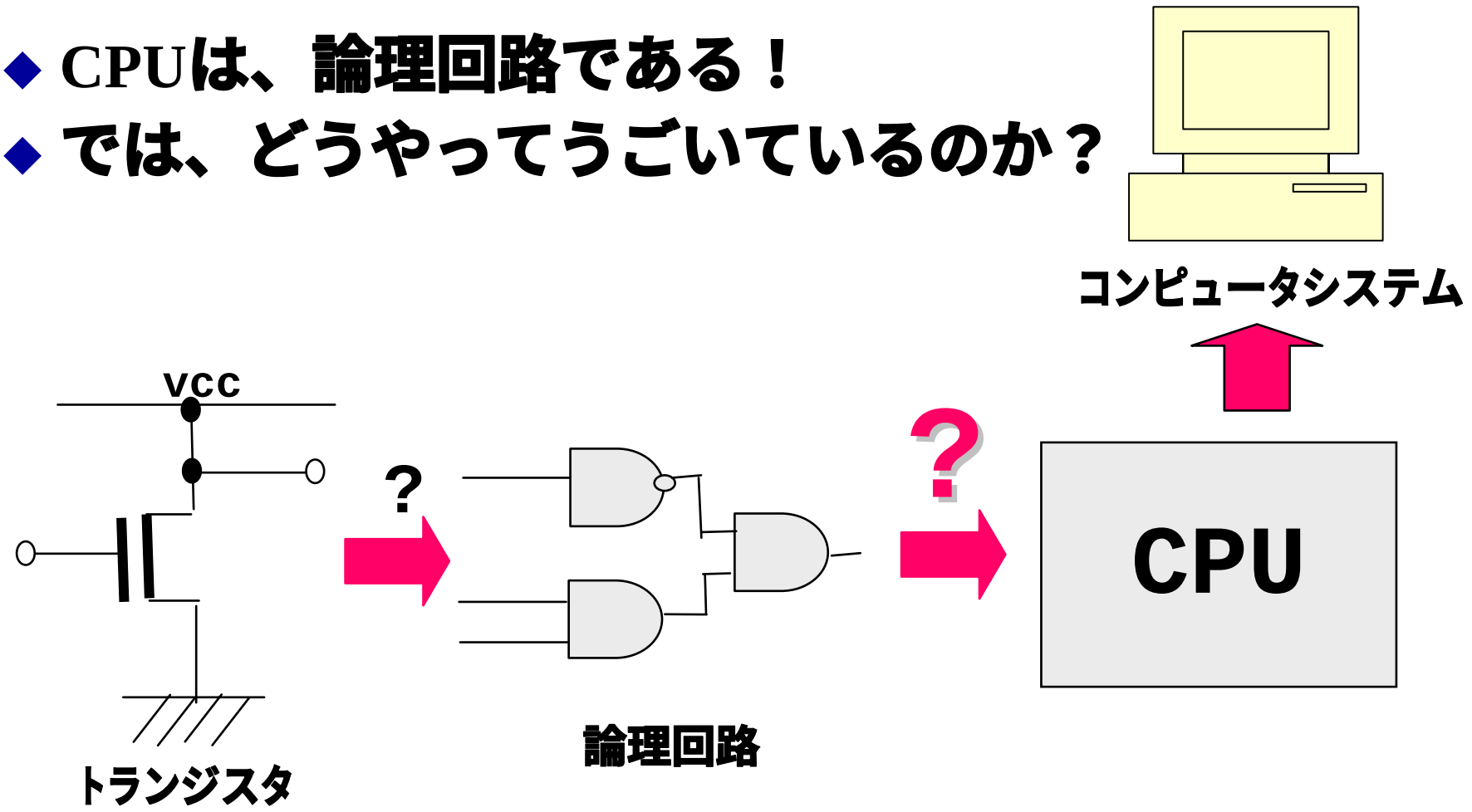
この講義の目的

- この講義の目的は、機械語すなわちアセンブリ言語のプログラミングを通じて、コンピュータについての基礎的な仕組みを理解することです。
  - アセンブリ言語をすなわち機械語を直接プログラミングする機会はそれほど多くはありませんが、コンピュータがどのように動作しているかについて理解することは情報科学、コンピュータ科学を学ぶ上でもっとも基本的な事柄の一つです。

- ４ビットマイコンを理解しよう。

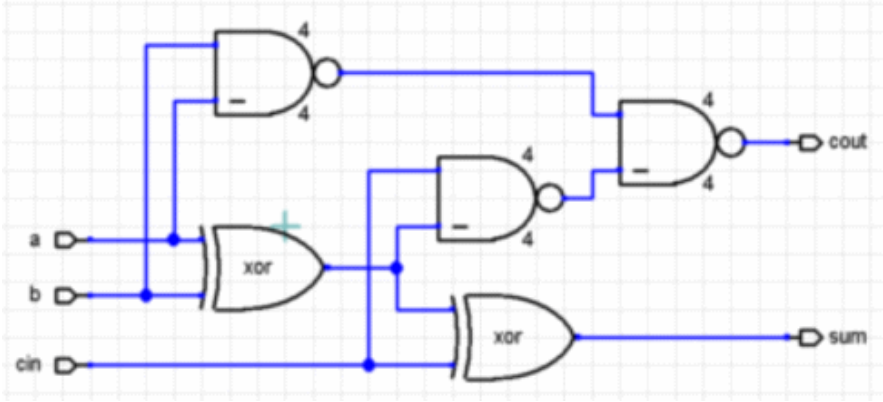
CPUと論理回路

- CPUは、論理回路である！
- では、どうやってうごいているのか？



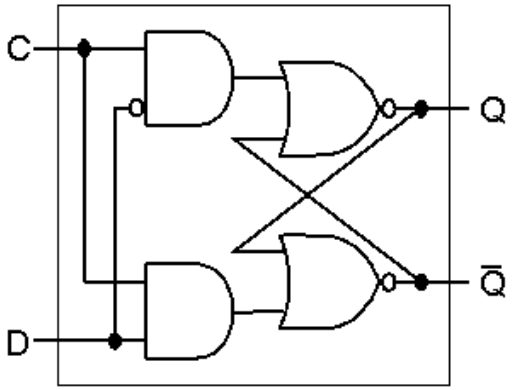
論理回路

- 組み合わせ回路
  - 入力により、出力が決まる
  - 論理素子AND, OR,NOTの組み合わせ
  - 加算器



full adder

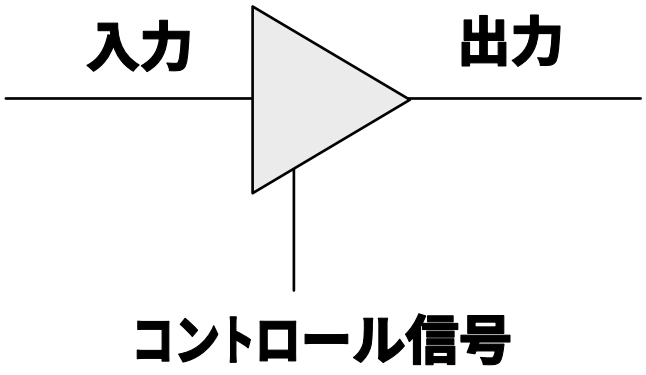
- 順序回路
  - 現在の出力が過去の入力の状態によって決まる回路
  - フリップフロップ
  - レジスタ、カウンタ、…
  - メモリ（電荷素子）

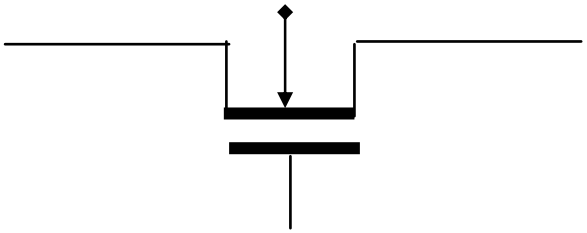


Dラッチ

論理回路

- バススイッチ（ゲート）
  - コントロール信号によって、出力を遮断する。





論理回路とコンピュータ

- コンピュータは、０と１の２進数で動作している。
- ０と１とは、電圧が高い、低い、あるいは、メモリでは電気がたまっている、たまっていない、といった２つの物理的な状態で表現されている。

- 例：加算器

コンピュータの基本的な構成

- コンピュータのもっとも基本的な要素は、メモリとプロセッサ（CPU)である。
  - メモリはプログラムやデータを格納する場所
  - プロセッサはそのメモリからプログラムやデータを読み出して、プログラムを実行しています。
    - 指令する部分：プログラムを解釈（？）して指令する
    - 演算する部分：足し算や掛け算をする部分

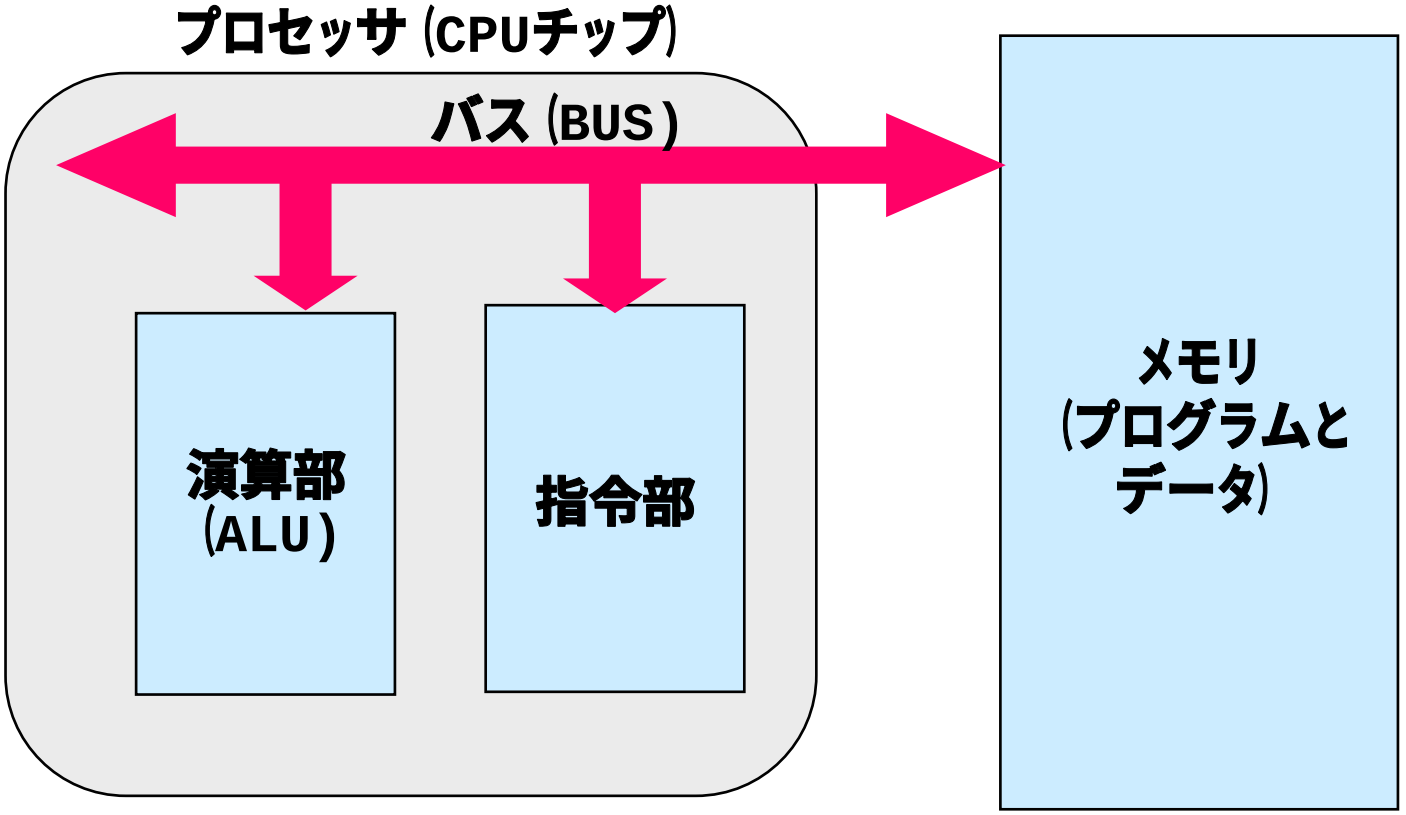
- プログラムとデータをメモリにいれて、プロセッサがメモリから読み出して実行する方式を、ストアードプログラム方式という。

現在のコンピュータのもっとも重要な基本的な概念

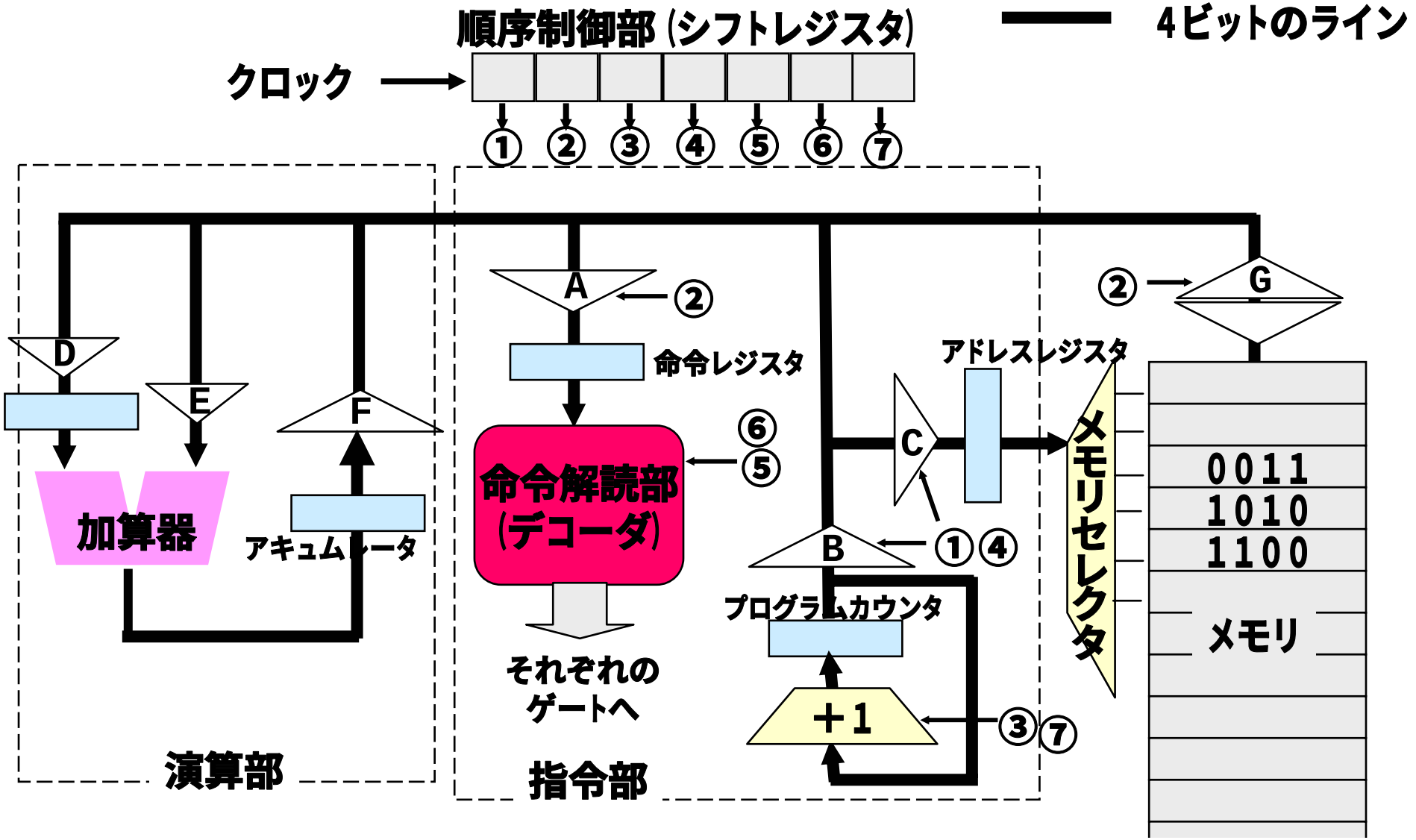
プログラムを実行するプログラムがつくれる⇒システム

コンピュータの基本的な構成

- 記憶（メモリ）、指令、演算



簡単な４ビットプロセッサ



説明

- 命令やデータのとおり道（実際は電線）を「バス」という。
  - データのとおり道はデータバス、
  - メモリにどのデータを読み出すかを伝えるバスをアドレスバス
- メモリの横にある「番人」（セレクタという）に接続されているのがアドレスバスで、操作するメモリを指定していま三角でしめされているのがゲート。信号の流れを制御する。
- このゲートを制御する信号は、現在の命令（０と１の組み合わせ）から、命令解読部（デコード）で作られる

- プロセッサの中にも一時的にデータを格納するメモリ（のようなもの）がある。レジスタと呼ばれる。そのいくつかはプログラムからは見えない（例えば、現在の命令を保持している命令レジスタや読み出すメモリの番地を保持しているアドレスレジスタなど）
- 実行するプログラムの番地を保持しているレジスタをプログラムカウンタという
- 演算の一時的な結果を保持するレジスタをアキュムレータと呼ぶことがある。実際のプロセッサではこのようなレジスタが複数ある。

命令コード（機械語）

- メモリ上にあるプログラムのそれぞれの命令は、動作とその対象からなる。
  - 動作を指定するのが、命令コード（オプコード）
  - 対象をオペランドという
  - 実際のマシンではオペランドのない命令もある
- このマシンでは、２ワードであらわす

0001	X
0010	Y
0011	Z
0100	W

Xを足される数に設定しなさい（LOADI）

設定されている足される数に
Yを足しなさい（ADDI）

Zのアドレスに結果を入れなさい（STORE）

W番地の中身を、足される数に設定しなさい
（LOAD）

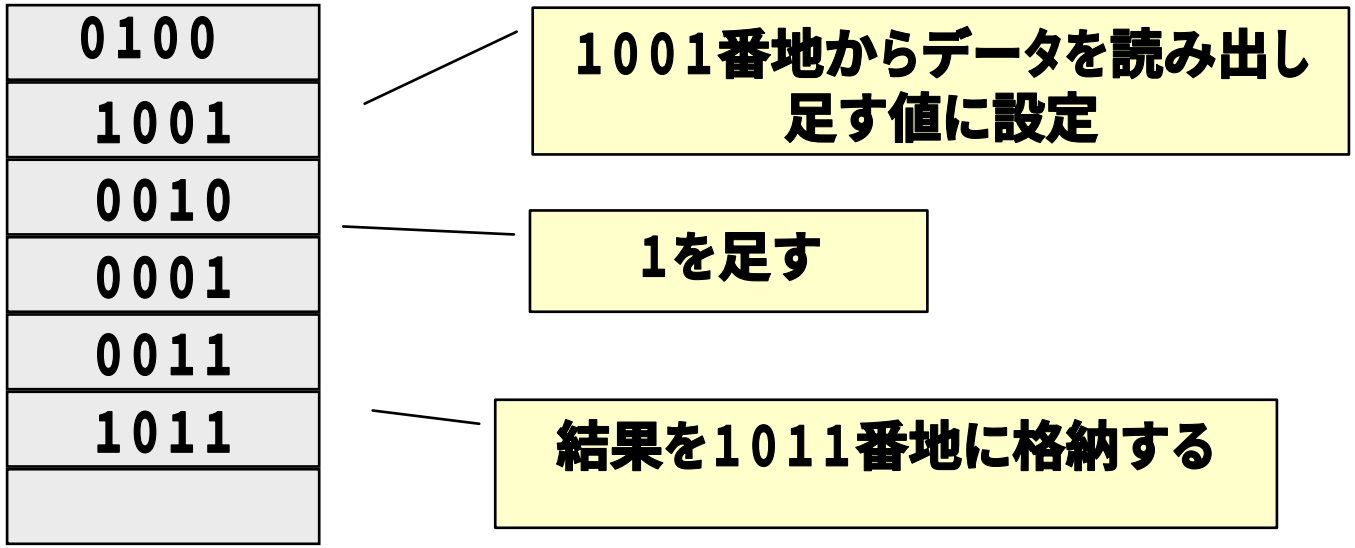
どのように実行されるか

- クロック信号が入力されると、順序制御部から①から⑥までの信号が順番に送られる
  - ①　プログラムカウンタをアドレスレジスタに設定
  - ②　命令コードのフェッチ
  - ③　プログラムカウンタを１つあげる（オペランドを読む準備）
  - ④　プログラムカウンタをアドレスレジスタに設定
  - ⑤　命令コードの解読と実行１
  - ⑥　命令コードの解読と実行２
  - ⑦プログラムカウンタを１つあげる（次の命令を読む準備）
- これが、コンピュータの速度を決定する

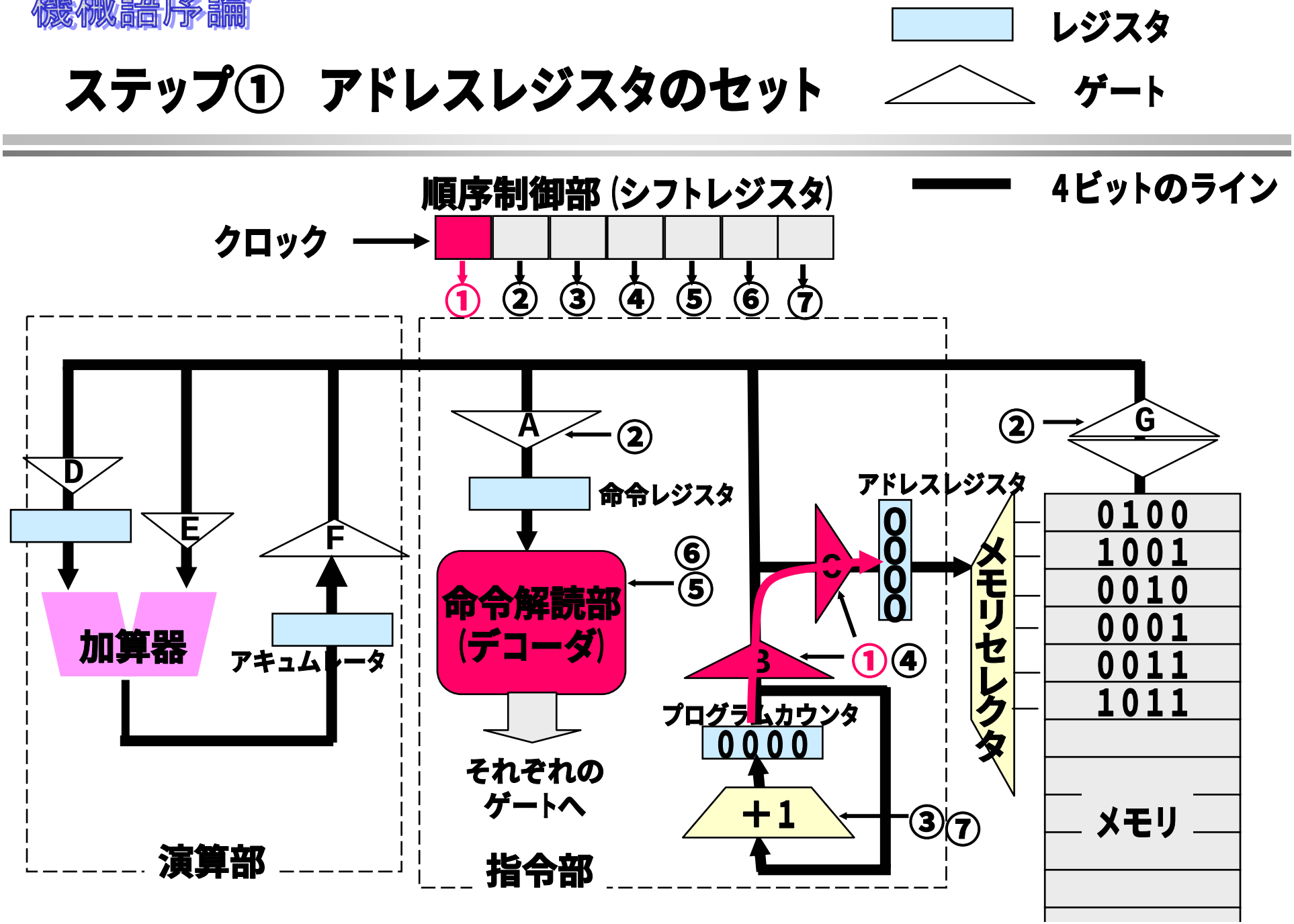
- この単純なプロセッサでは、⑤⑥以外は同じパターンでゲートを開け閉めするが、⑤⑥だけは現在の命令から、デコード部で生成された信号でゲートを開け閉めする
  - このデコード部は組み合わせ回路である！

プログラム例

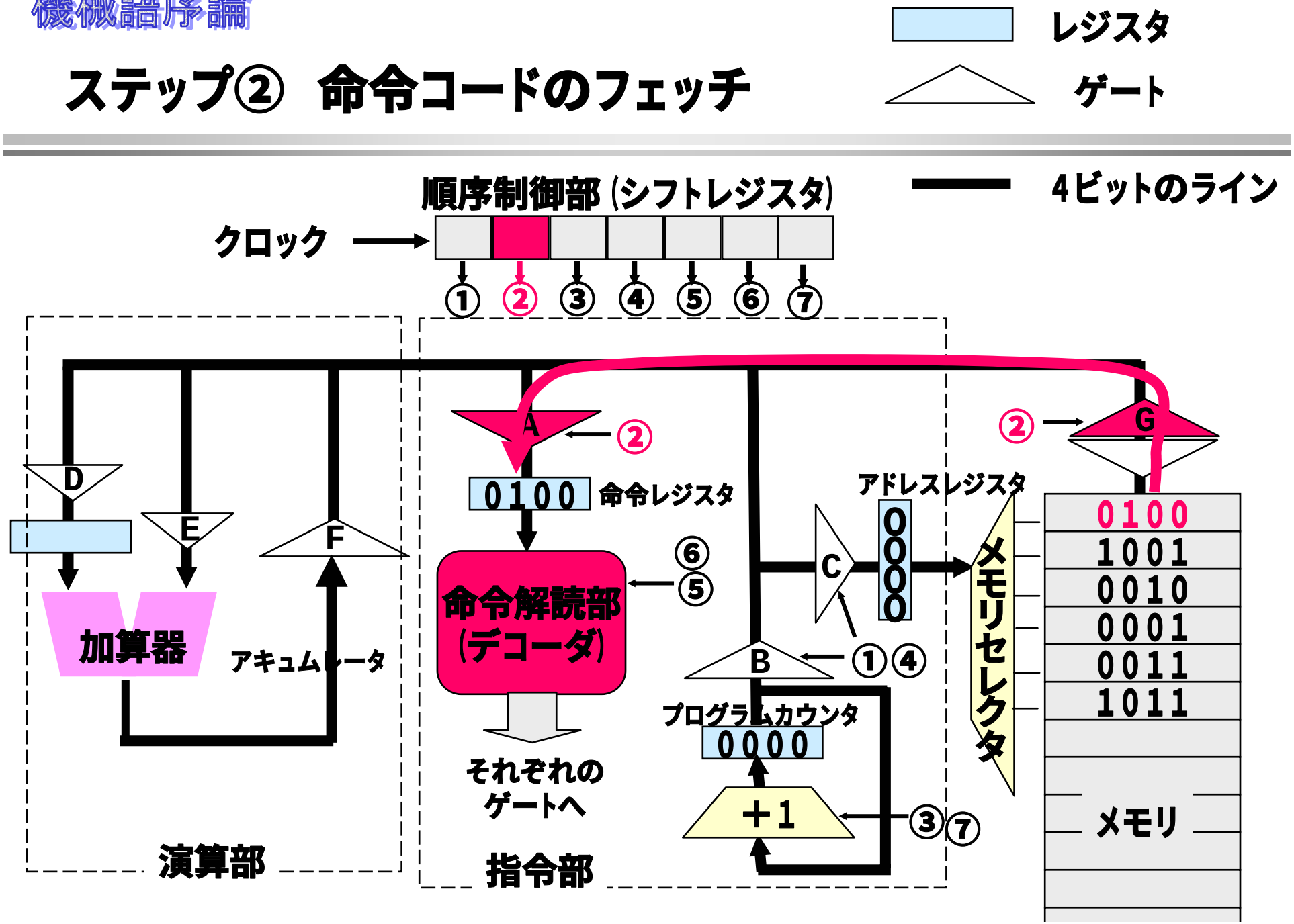
- ここで、番地１００１からデータを読み出し、１を加えて、番地１０１１に格納するというプログラムを考える



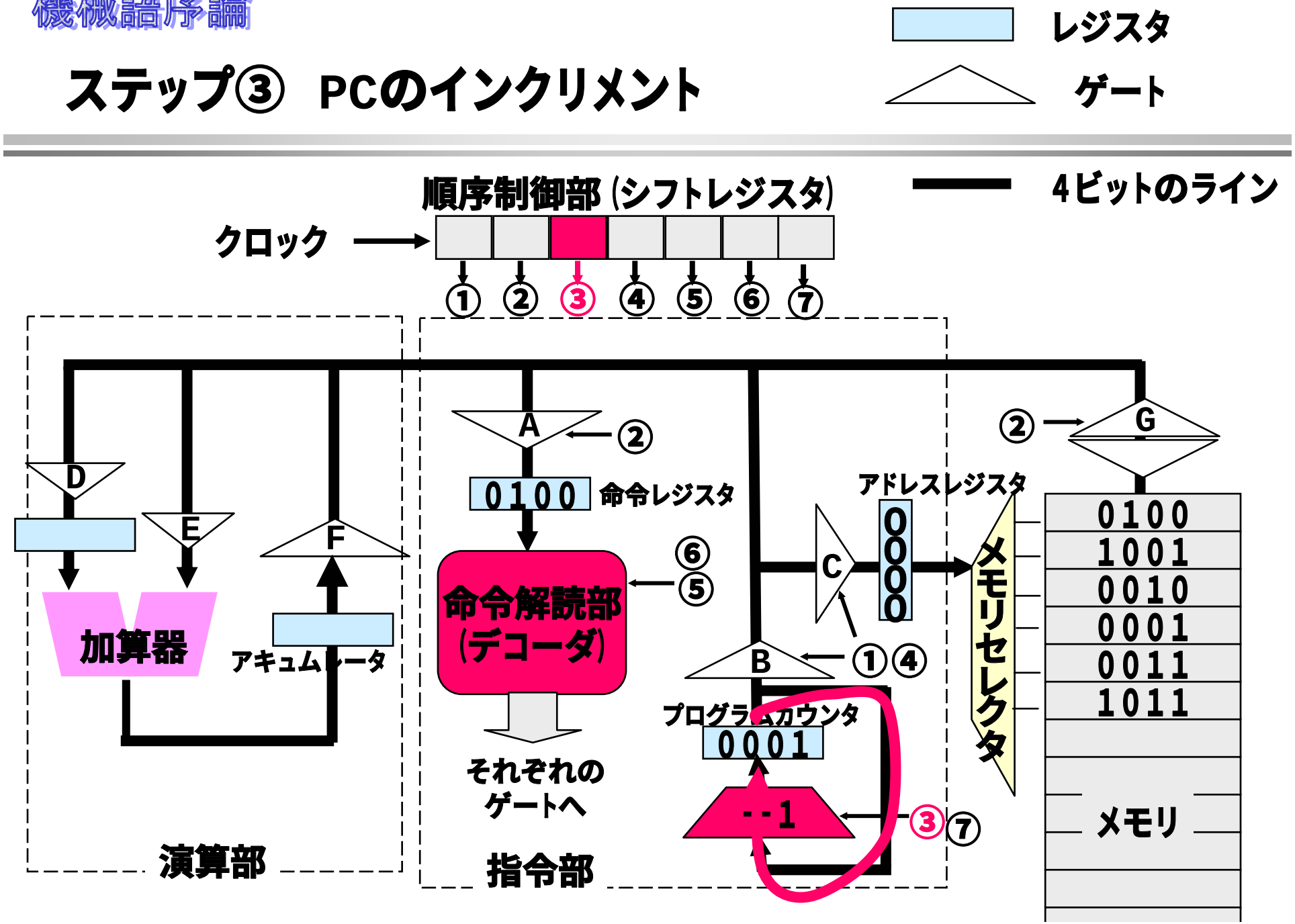
ステップ①　アドレスレジスタのセット



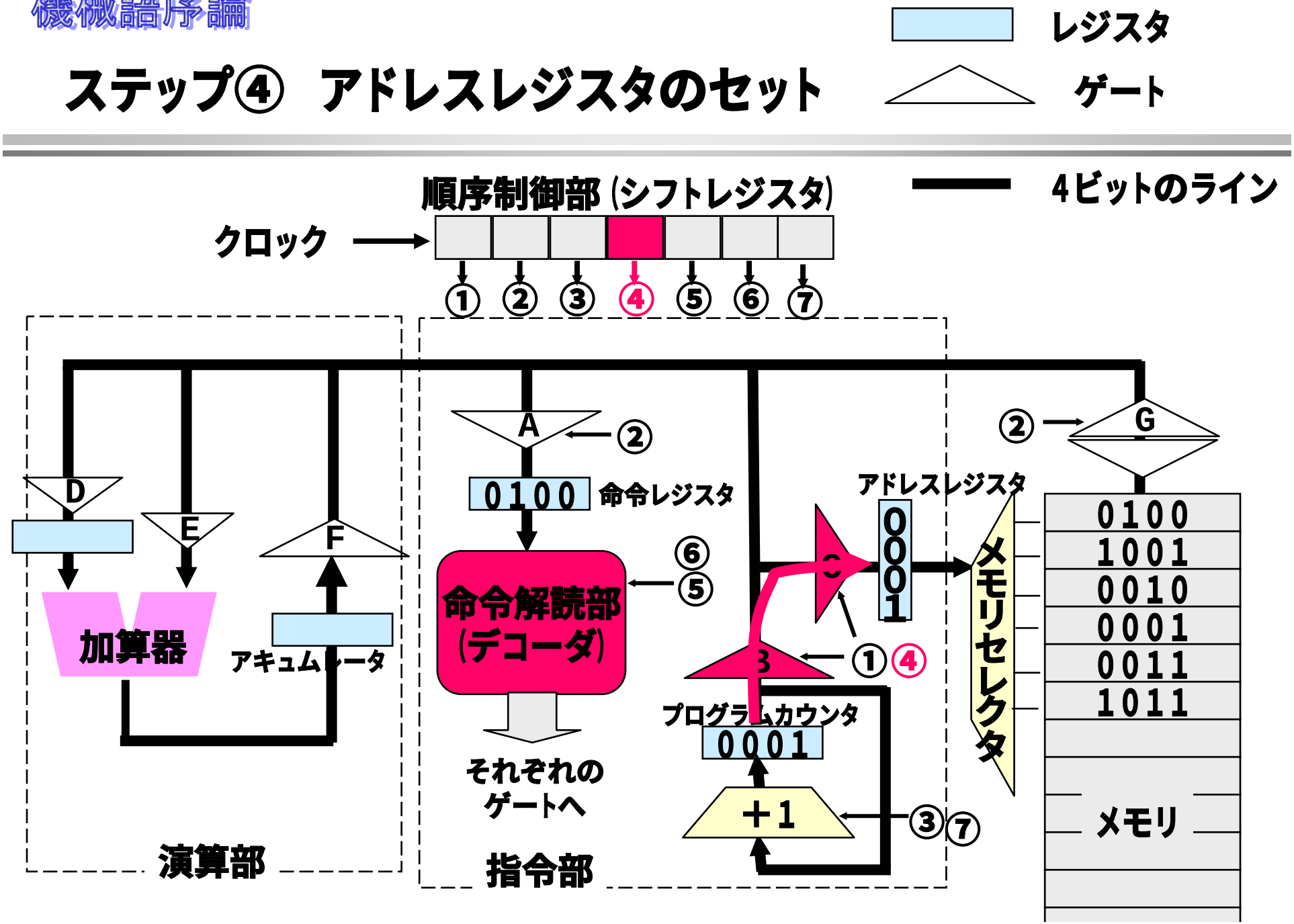
ステップ②　命令コードのフェッチ



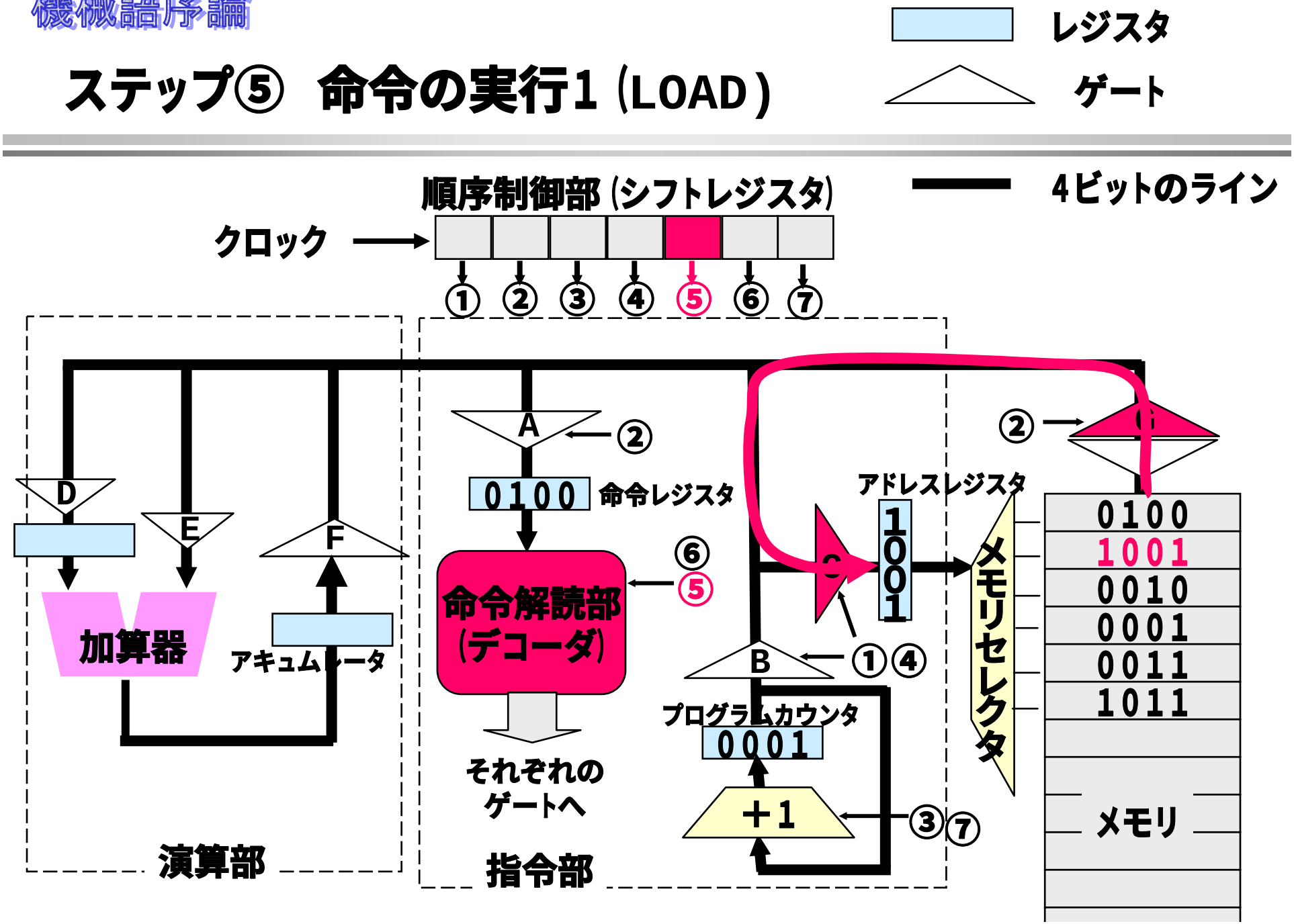
ステップ③　PCのインクリメント



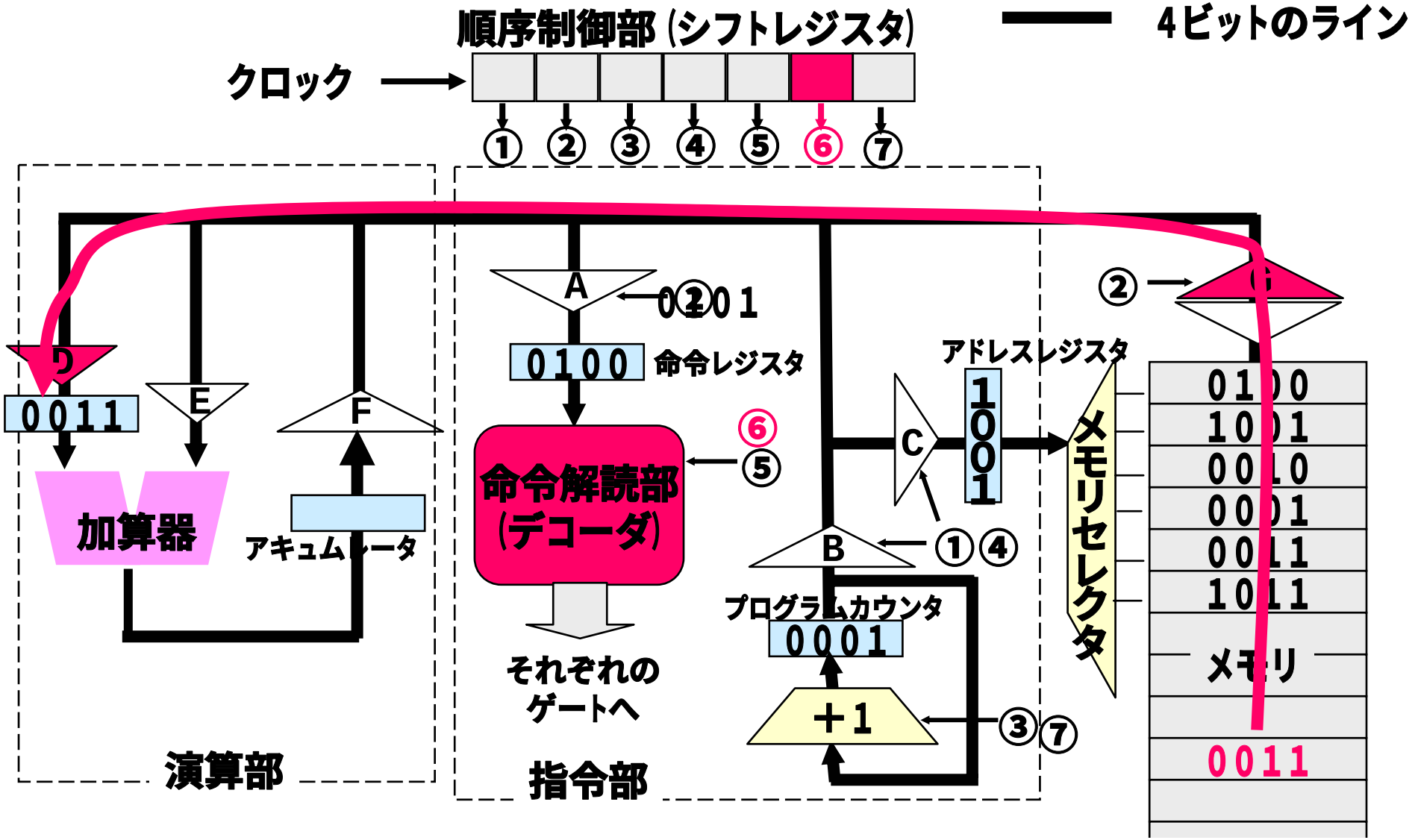
ステップ④　アドレスレジスタのセット



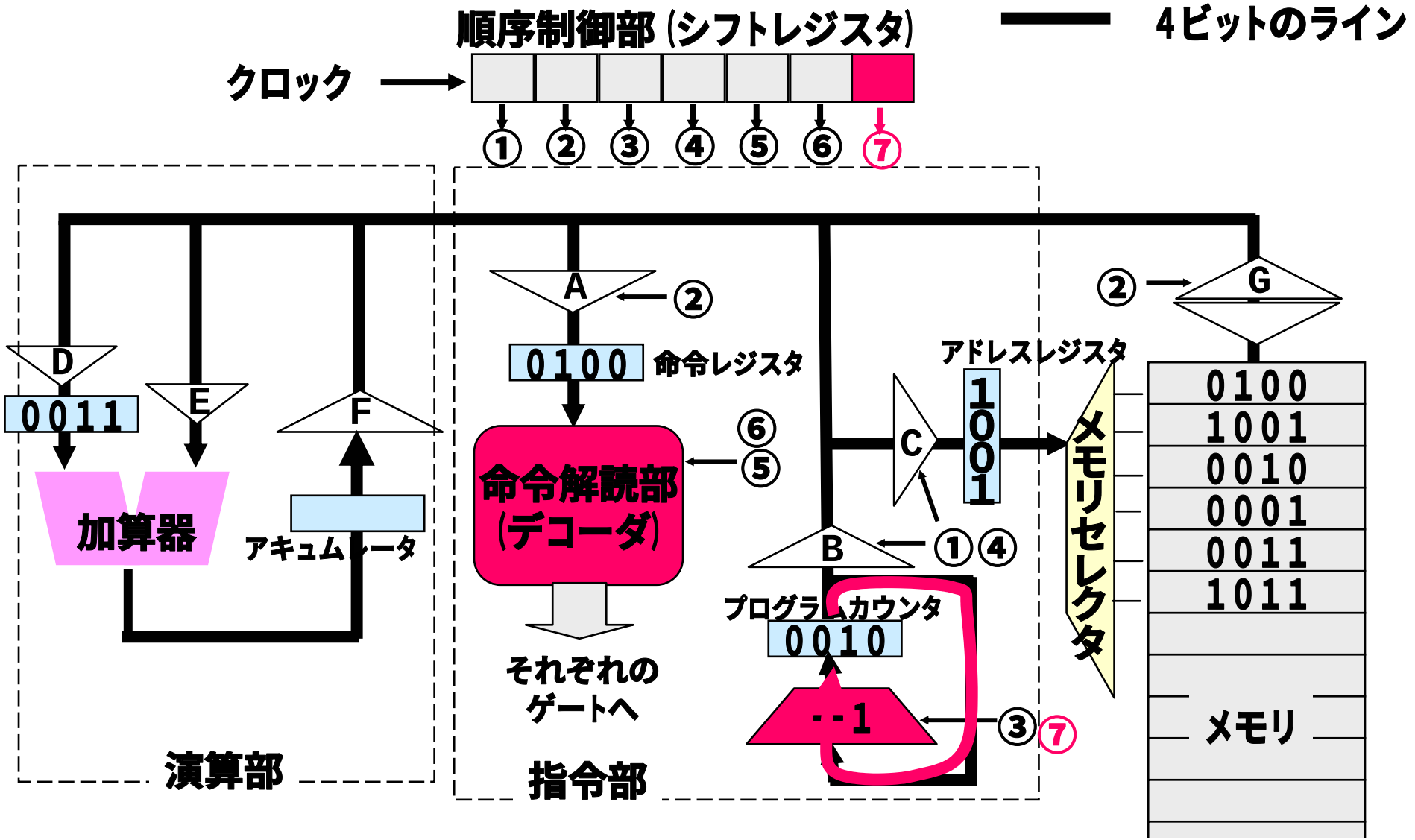
ステップ⑤　命令の実行1（LOAD）



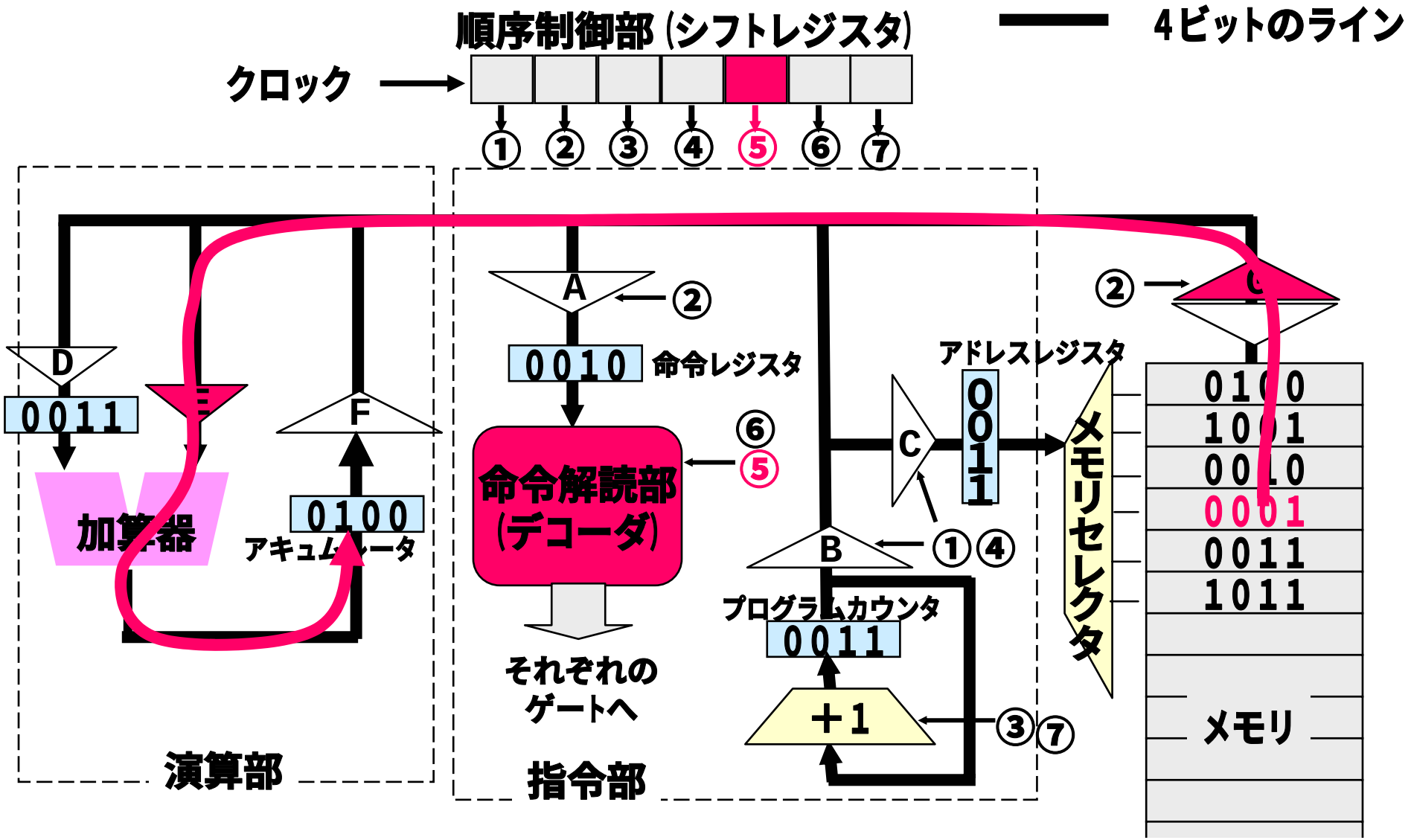
ステップ⑥　命令の実行2　（LOAD）



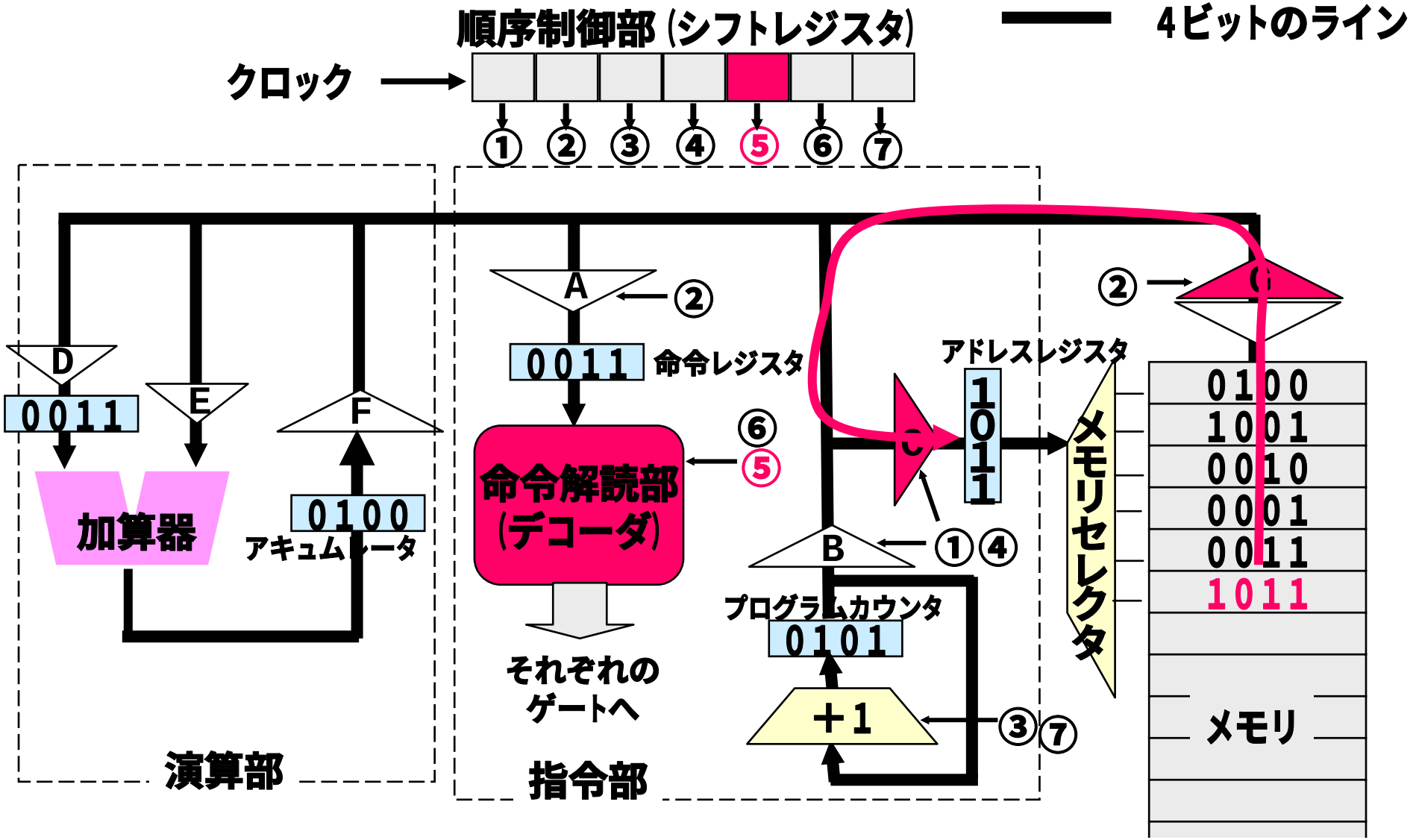
ステップ⑦　PCのインクリメント



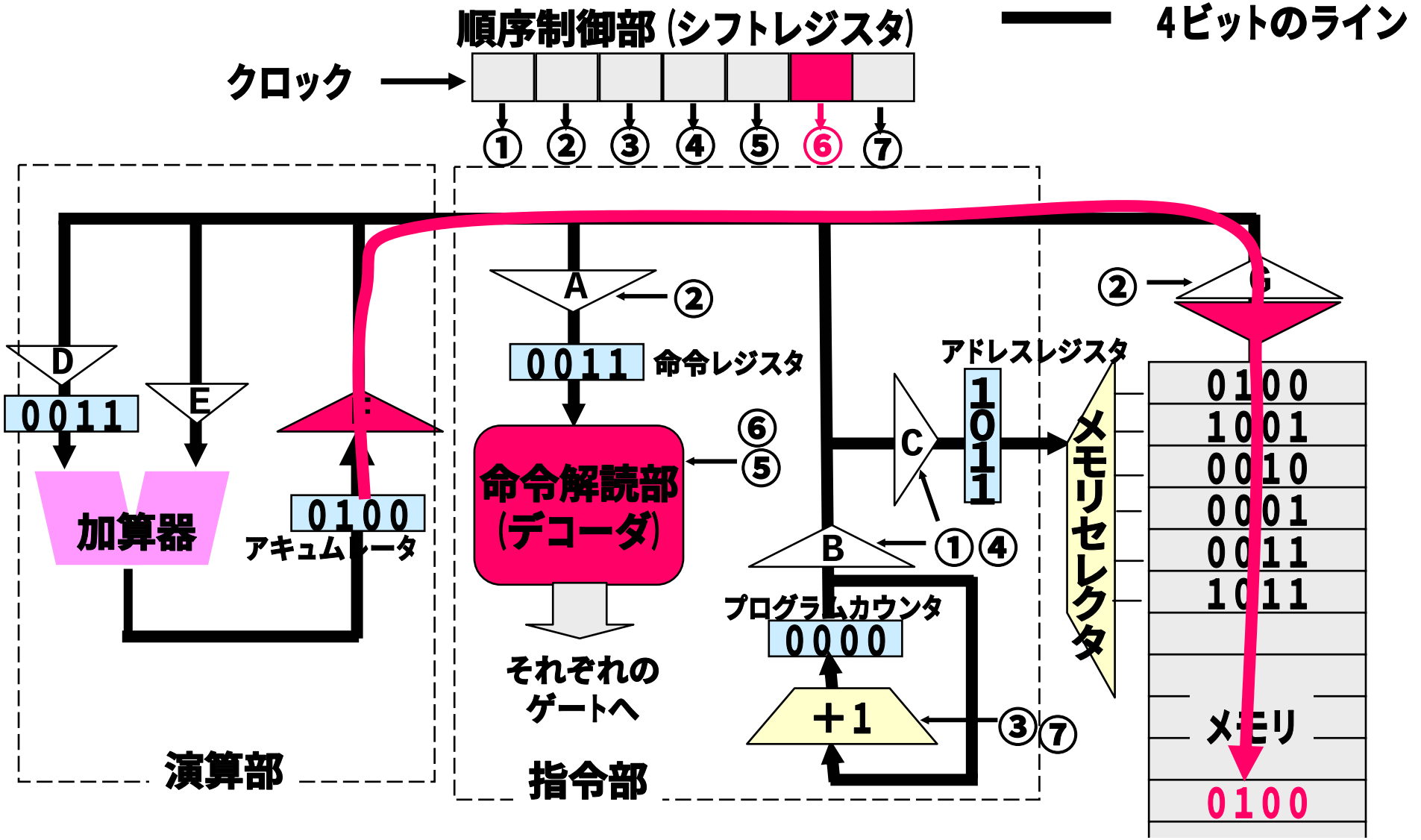
ステップ⑤　加算の実行


レジスタ
ゲート
4ビットのライン
順序制御部 (シフトレジスタ)
クロック
① ② ③ ④ ⑤ ⑥ ⑦
D
0011
E
F
0100
加算器
アキュムレータ
演算部
A
②
0010
命令レジスタ
⑥
⑤
命令解読部
(デコーダ)
それぞれの
ゲートへ
C
アドレスレジスタ
0011
B
①④
プログラムカウンタ
0011
+1
③⑦
指令部
②
メモリセレクタ
0100
1001
0010
0001
0011
1011
メモリ

ステップ⑤　STOREの実行1

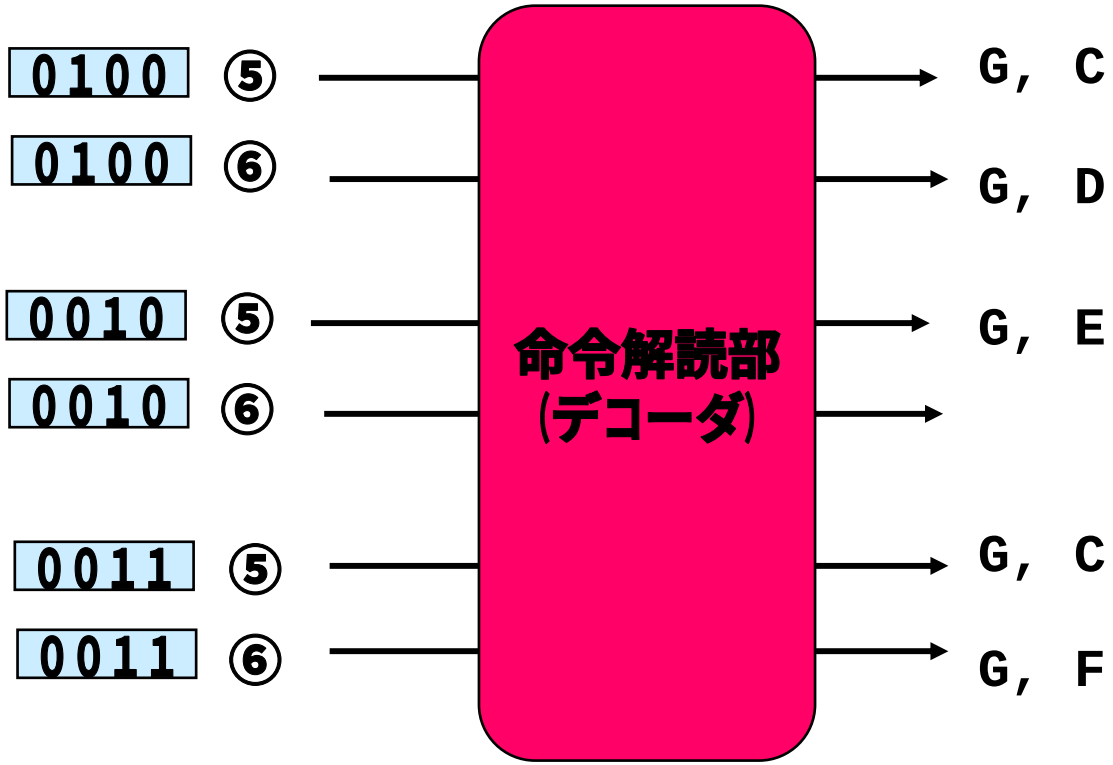


ステップ⑥　STOREの実行2



命令解読部（デコーダ）の働き

- 命令のデコーダは、基本的には組み合わせ回路

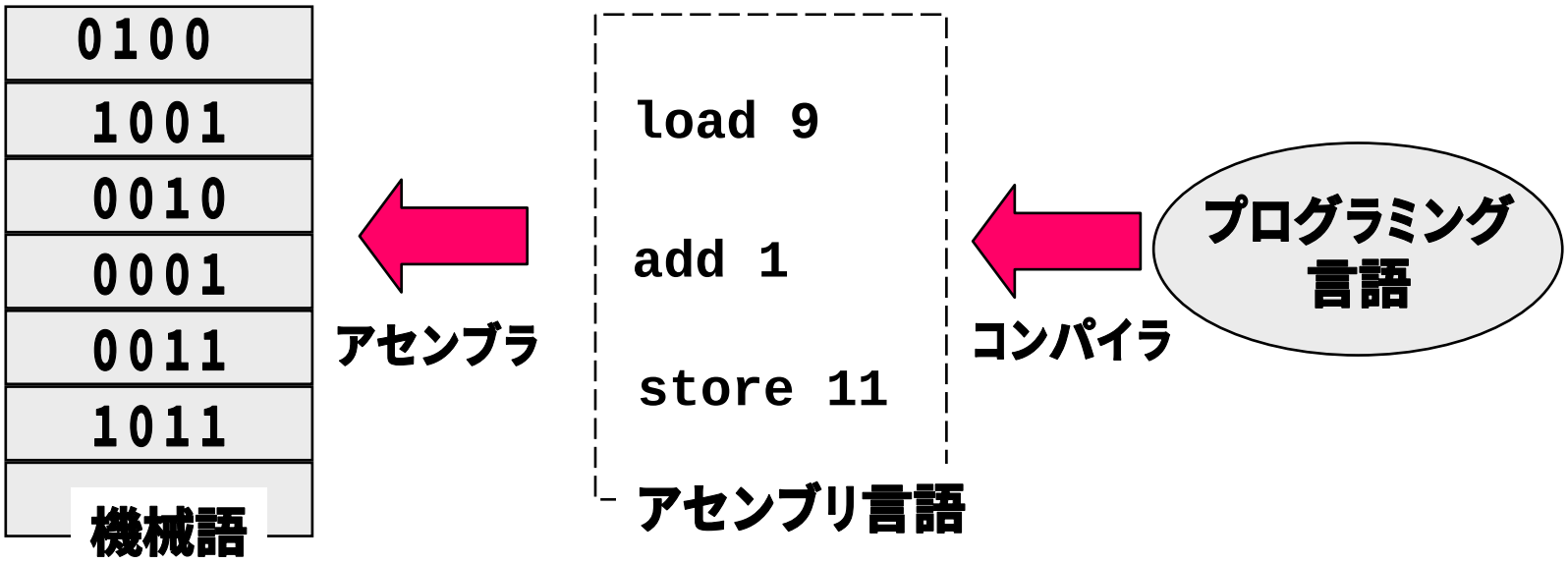


本当のマイクロプロセッサでは…

- 制御を変えるにはどうすればいい？(jump命令、条件分岐)
- 実際には、各ステップで同時に実行できるところがある。
  - たとえば、実行とプログラムカウンタのインクリメントを同時に実行
- プロセッサの実行フェーズ
  - 命令フェッチ　(IF : instruction fetch)
  - 命令デコード　(ID: instruction decode)
  - 命令実行　(EX: execute)
  - 結果の書き込み(WB: write back)
  - …
- レジスタがたくさんある。
  - 汎用レジスタ（整数とアドレス）
  - 浮動少数点レジスタ
- キャッシュメモリがある。
- オペレーティングシステムのいろいろな機能
  - 仮想記憶、割り込み、入出力

アセンブリ言語とコンパイラ

- マシン語をそのものでは０１のパターン、つまり数字ですので、これをつかってプログラミングするのは人間にとって非常に面倒な作業になる
- 基本的に、マシン語に１対１に対応するように記号を使って表記したのがアセンブリ言語
- オプコードを表す記号をニーモニック という
- コンパイラは、プログラミング言語をアセンブラ言語に翻訳する。



機械語序論

- アセンブリ言語で記述されたプログラムは、アセンブラによって、機械語に翻訳（変換）されます。
- アセンブリ言語は基本的には機械語と１対１なので、アセンブリ言語で書かれたプログラムは機械語で書かれたプログラムとは同じものとして考えることができます。
- Ｃコンパイラでは、－Ｓ　オプションをつけてコンパイルすると、.sというファイルができるはずです。どんなコードができているかをみてください。
  - % cc –S t.c

x86 (IA32)　プロセッサ

- この講義では、インテルのx86ファミリーと呼ばれるプロセッサを題材に進めていきます。
  - このプロセッサは学類の計算機で使われているプロセッサであり、普通のＰＣに使われているプロセッサでもあります。
  - x86プロセッサはいろいろな種類がありましたが、この講義では80586移行のいわゆるPetinumプロセッサファミリを対象とします。

x86 (IA32)　プロセッサの特徴

- ＣＩＳＣ(Complex instruction set computer)であるが、内部的に簡単な命令に分解して実行する機能を持ち、非常に高速化されている。
- 論理的なアドレス空間は３２ビット（最近は６４）
- レジスタは、ＰＣ（プログラムカウンタ）のほか、３２ビットの汎用レジスタとして、eax, ebx, ecx, edx, esi, edi, esp, ebp の８個のレジスタがある。このうち、espは、スタックポインタ、ebpはベースレジスタと名づけられ、ソフトウエア的につかい方が決まっている。
- メモリアクセスする場合には、豊富なメモリアクセスモードが使える。
- 浮動小数点レジスタは８個で、スタック状に使う。

- このプロセッサは歴史的な経緯を引きずっており、非常に複雑な命令セットになっていますが、この講義では、必要な部分のみを使ってプログラミングすることにします。

- 詳しく知りたい方は、Intelから出されている「Pentium ファミリー　デベロッパーズマニュアル（下）アーキテクチャとプログラミングマニュアル」などを参照してください。

次回

- アセンブリ言語のプログラミング環境について解説します。
  - アセンブリ言語のプログラムの実行の確認やデバックのために、gnu debugger 、gdbを使いますので、その使い方についても解説します。