コンピュータシステムB -ソフトウェアを中心に -

#8 コンパイル・プログラムの実行

Yutaka Yasuda

プログラミング

- 「1から10までの数を足した結果を出せ」
  これはコンピュータにとって「難しすぎる」指示
- 手順の明確化（アルゴリズムを得る）
  「Xを 1 から 10 まで変化させ、毎回 Y に繰り込め」
- プログラムとは何か
  目的に対して「何をどう処理するか」を詳述したもの
  アルゴリズムをプログラミング言語で書き下したもの

手順をどのように書くか

- コンピュータは日本語を理解できない

  「Xを 1 から 10 まで変化させ、それを毎回 Y に繰り込む」のもコンピュータには複雑すぎる

  コンピュータが理解できる言語で書きくだす必要がある

C言語での例

```
Y=0;
for(X=1;X<=10;X++)  {
   Y=Y+X;
}
```

Yははじめ 0 だと設定している

1から10まで変化させるということを、「1 からはじめて 10 以下の場合は終わりまでの処理を行い、1 加算してもう一度繰り返し」という表現で明記している

Yに増え続ける X を足したものを再び Y に代入

手順をどのように書くか

- しかしC言語でもハードウェアには直接理解できない

  さらに単純な処理に分解しなければ

  CPU(例は Motorola 68000)向けのアセンブリ言語で記述

```
        st 0,$1004
        mov 1,%o0
        st %o0,$1000
.LL2:
        ld $1000,%o0
        cmp %o0,11
        bgt .LL3
        ld $1004,%o0
        ld $1000,%o1
        add %o0,%o1,%o0
        st %o0,$1004
        ld $1000,%o1
        add %o1,1,%o0
        mov %o0,%o1
```

C言語での例

```
Y=0;
for(X=1;X<=10;X++) {
   Y=Y+X;
}
```

手順をどのように書くか

- アセンブリ言語でもまだ直接は理解できない。

  更に機械向けの言語＝機械語に直さなくては

  （以下は機械語の一部）

```
7400 5f5f 4354 4f52 5f4c 4953 545f 5f00
5f65 6e76 6972 6f6e 005f 656e 6400 5f47
4c4f 4241 4c5f 4f46 4653 4554 5f54 4142
4c45 5f00 6174 6578 6974 0065 7869 7400
....
```

二つの変換

人間側

1から10までの数を
足した結果を得る

人間が変換
（プログラミング）

```
Y=0;
for(X=1;X<=10;X++) {
   Y=Y+X;
}
```

機械的に変換できる
（コンパイラ）

```
02af93e8f
37de76e0a
4e3a2…
```

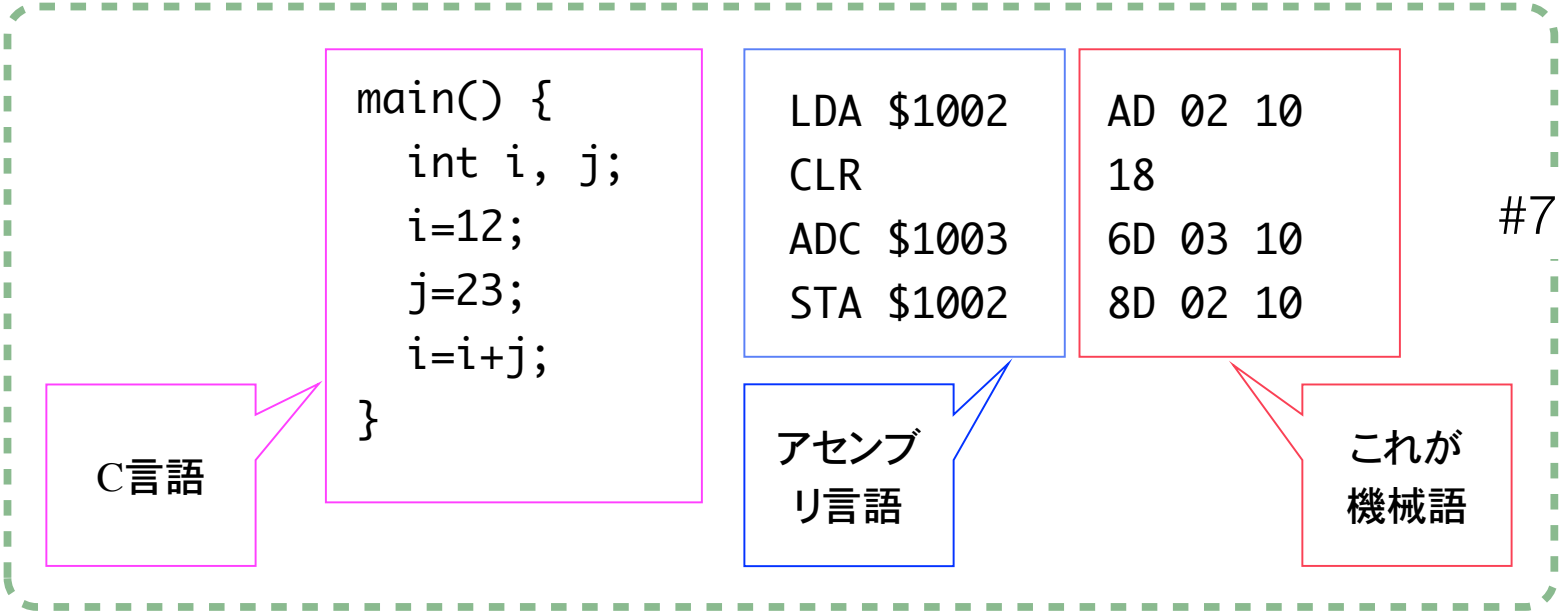
コンピュータ側

コンパイラとインタプリタ

- プログラミング言語から機械語へ

  CPUは機械語しか実行できない

  機械語によって等価な振舞いをさせなければ



#7より再掲

二つの手法：コンパイラとインタプリタ

コンパイラ

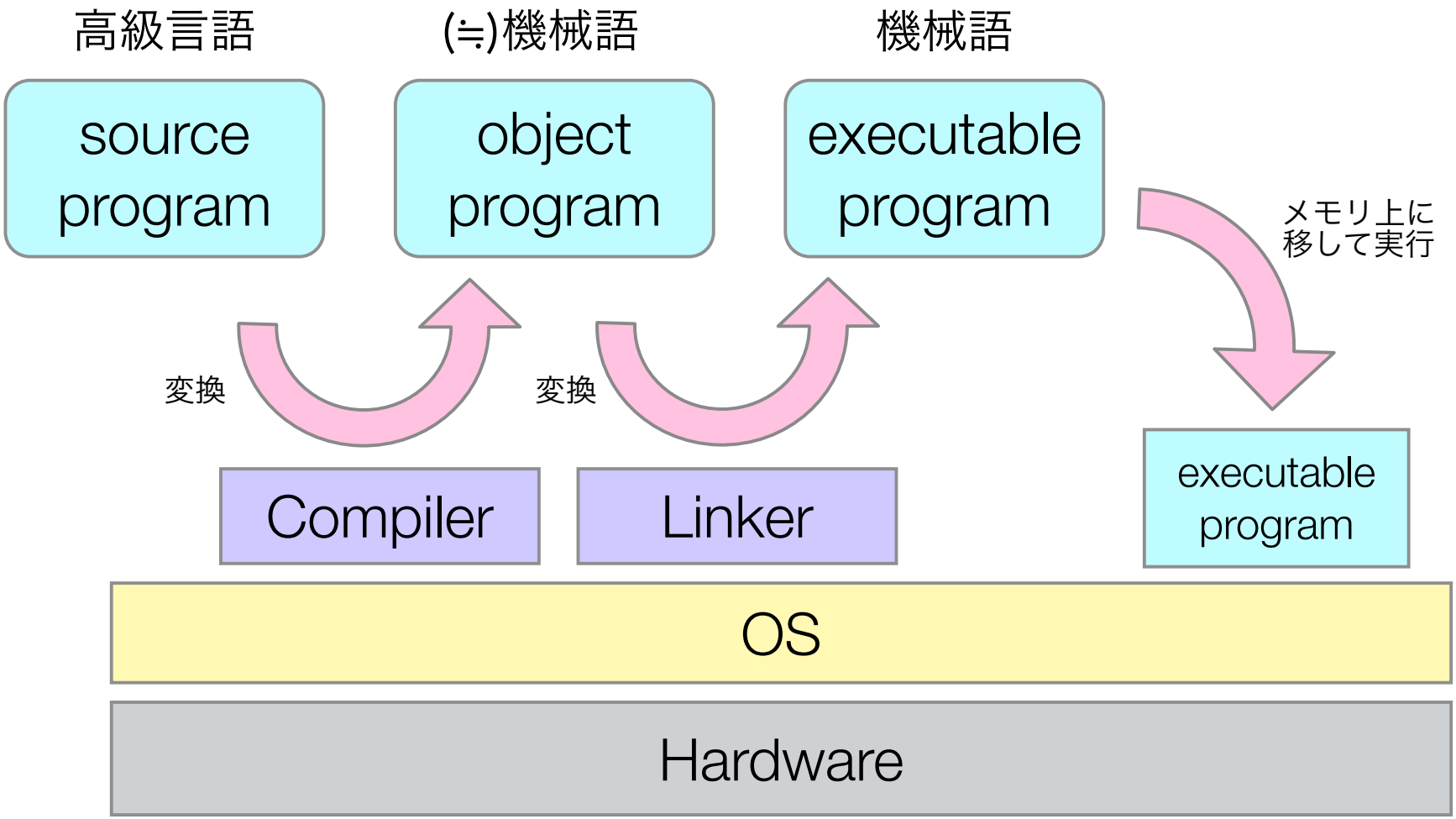
- Compiler = 編集、編纂
- 機械語に変換して実行

  変換前：原始プログラム (Source program)

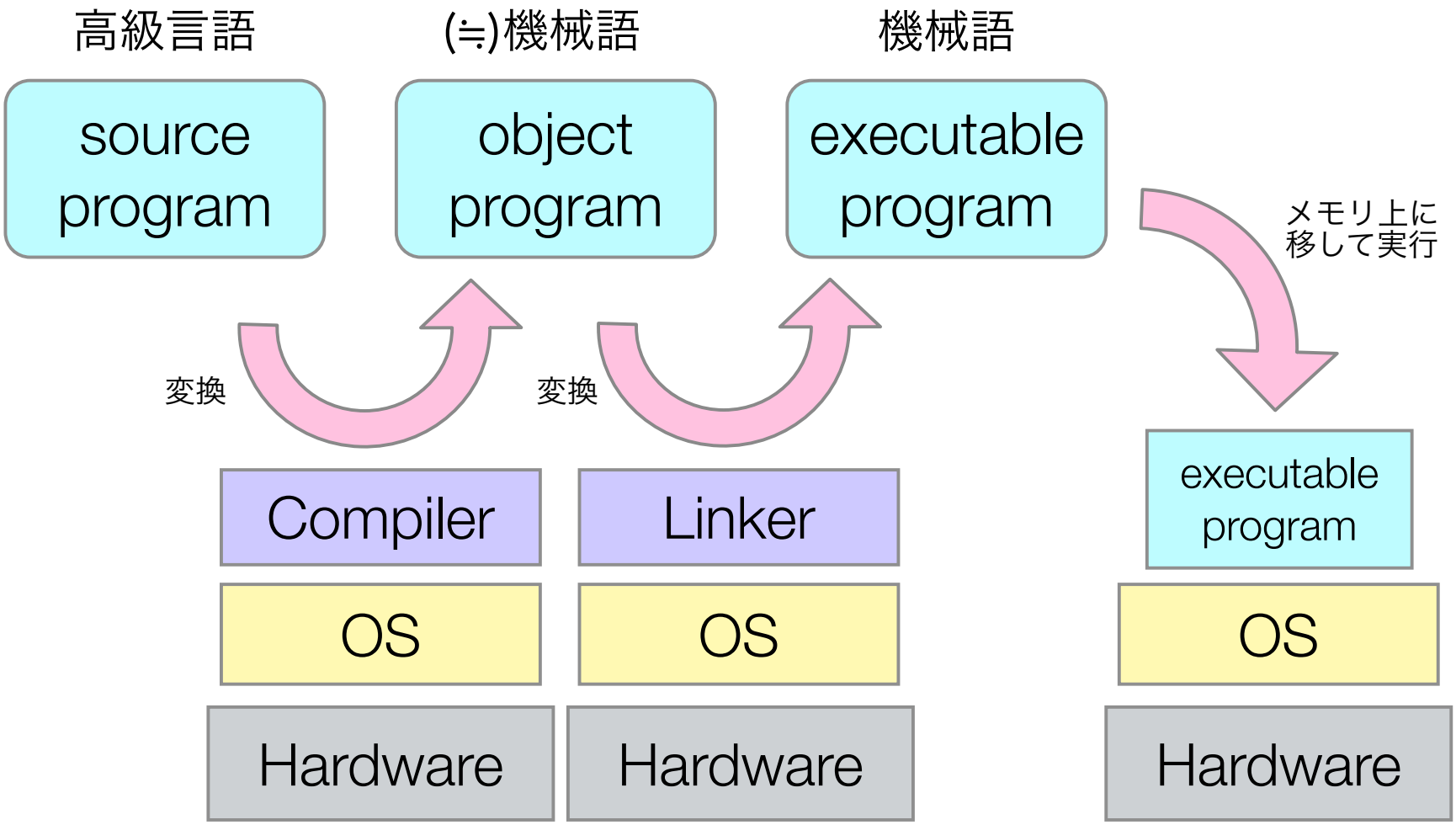
  変換後：目的プログラム (Object program)

- 多くの OS では目的プログラムをリンク (link) と呼ばれる処理を通してライブラリと結合し、実行可能プログラム (executable program) とし、これを実行する

コンパイラ


高級言語
(≒)機械語
機械語
source program
object program
executable program
メモリ上に移して実行
変換
変換
Compiler
Linker
executable program
OS
Hardware

コンパイラ



別々であっても構わない

コンパイラ

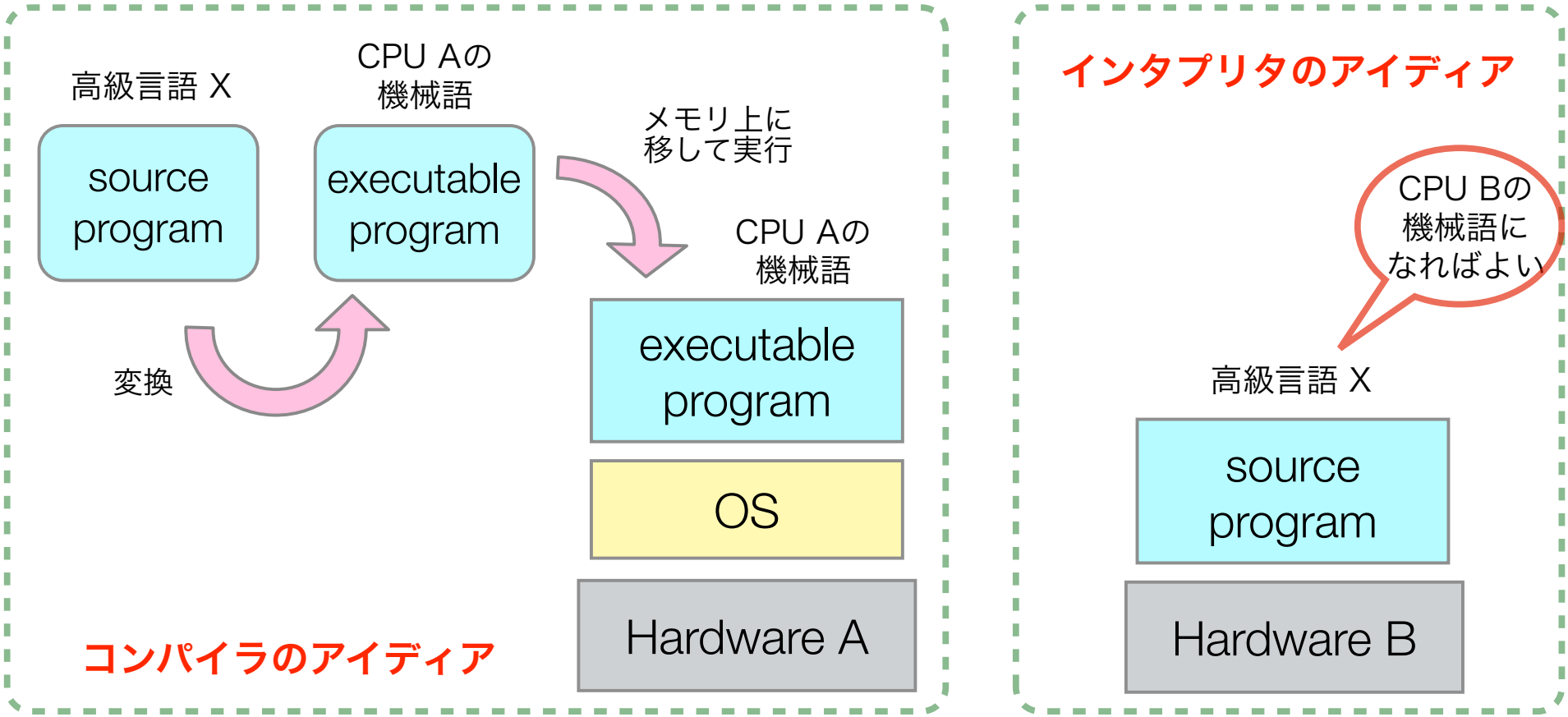
- 最適化の可能性
- 変換一回、実行多数回、では高効率
- 機密保持

    ソースが得られない場合が多い（逆変換不可）

- 商用ソフトウェアでこうした性質に合う場合多し

インタプリタ

- その言語を直接実行できる CPU をソフトウェアで作る
- 「仮想的なコンピュータ」を考える



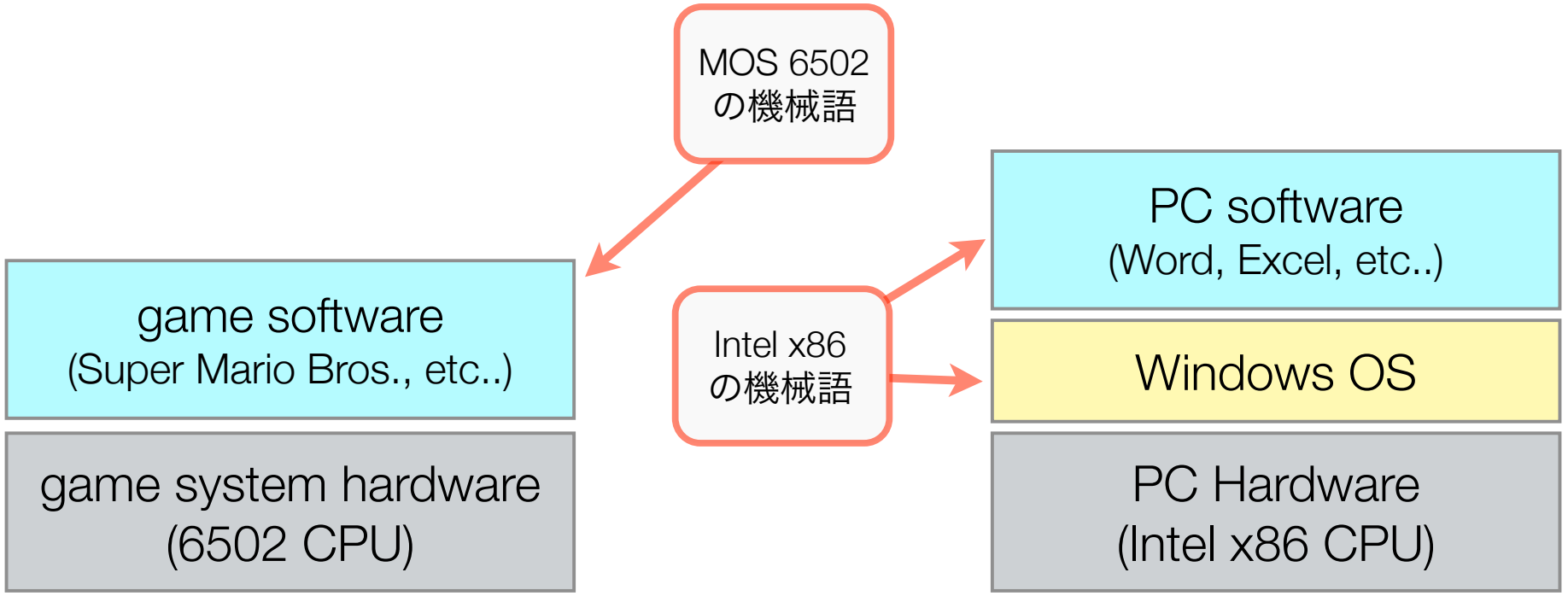
エミュレータ

- emulation：模倣
- あるCPU（or システム）を真似るソフトウェアを考える
- ゲームマシンの（ソフトウェアによる）エミュレータ

ゲームマシンのエミュレータ

ゲームソフトをゲームシステム
（実機）で実行する

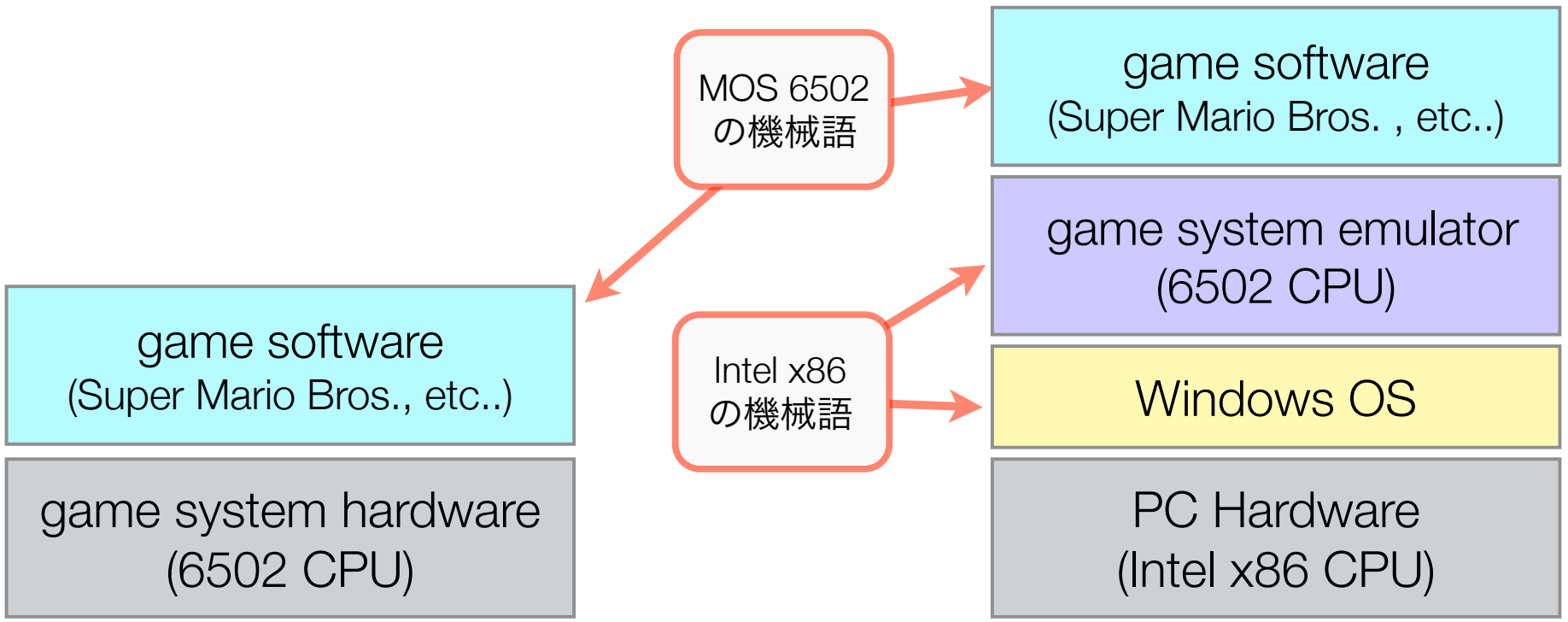
PCソフトをPC上で実行する



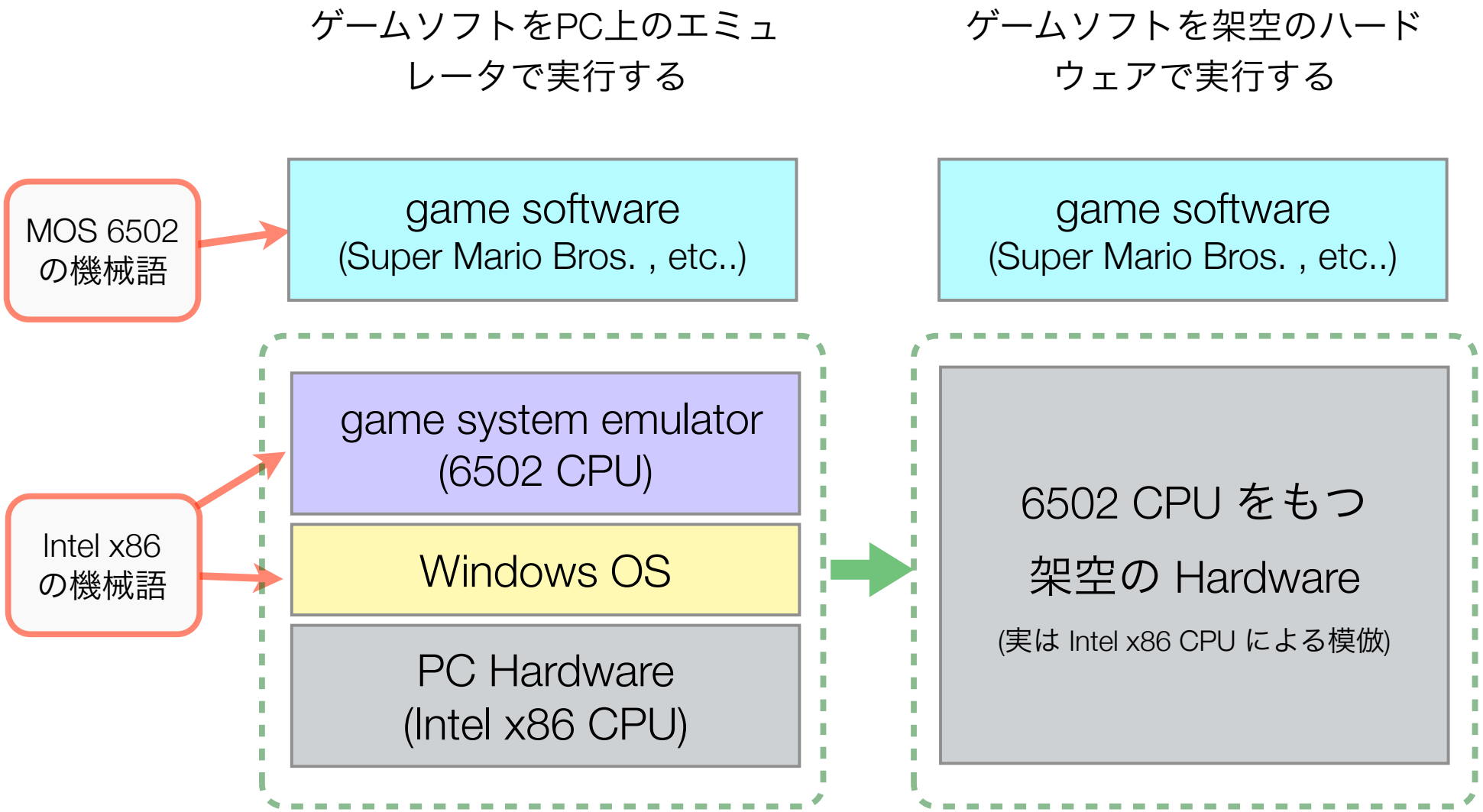
ゲームマシンのエミュレータ

ゲームソフトをゲームシステム（実機）で実行する

ゲームソフトをPC上のエミュレータで実行する

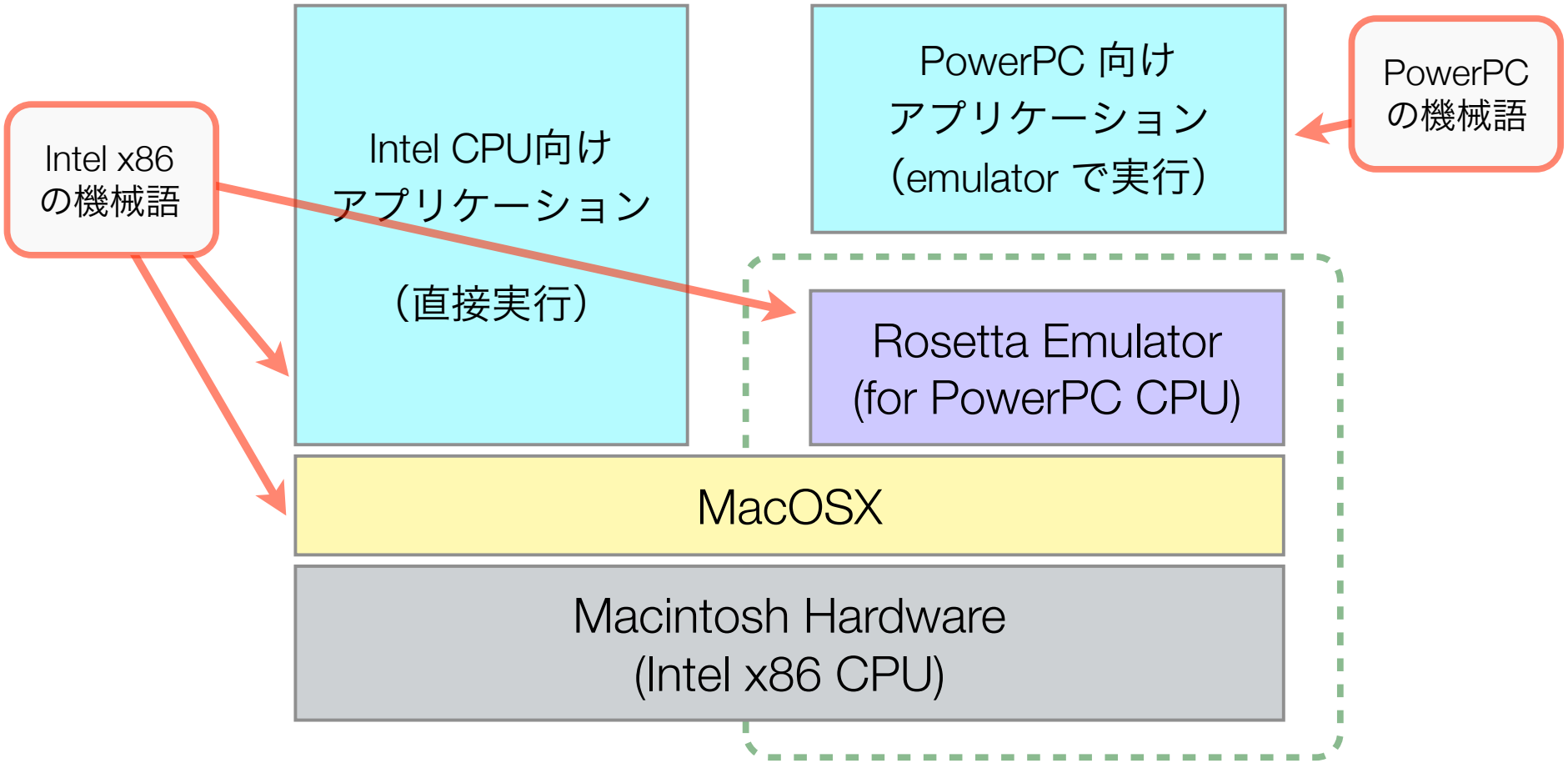


ゲームマシンのエミュレータ



MacintoshのRosetta

旧 CPU (PowerPC) 向けアプリケーションをエミュレータ上で動作させる

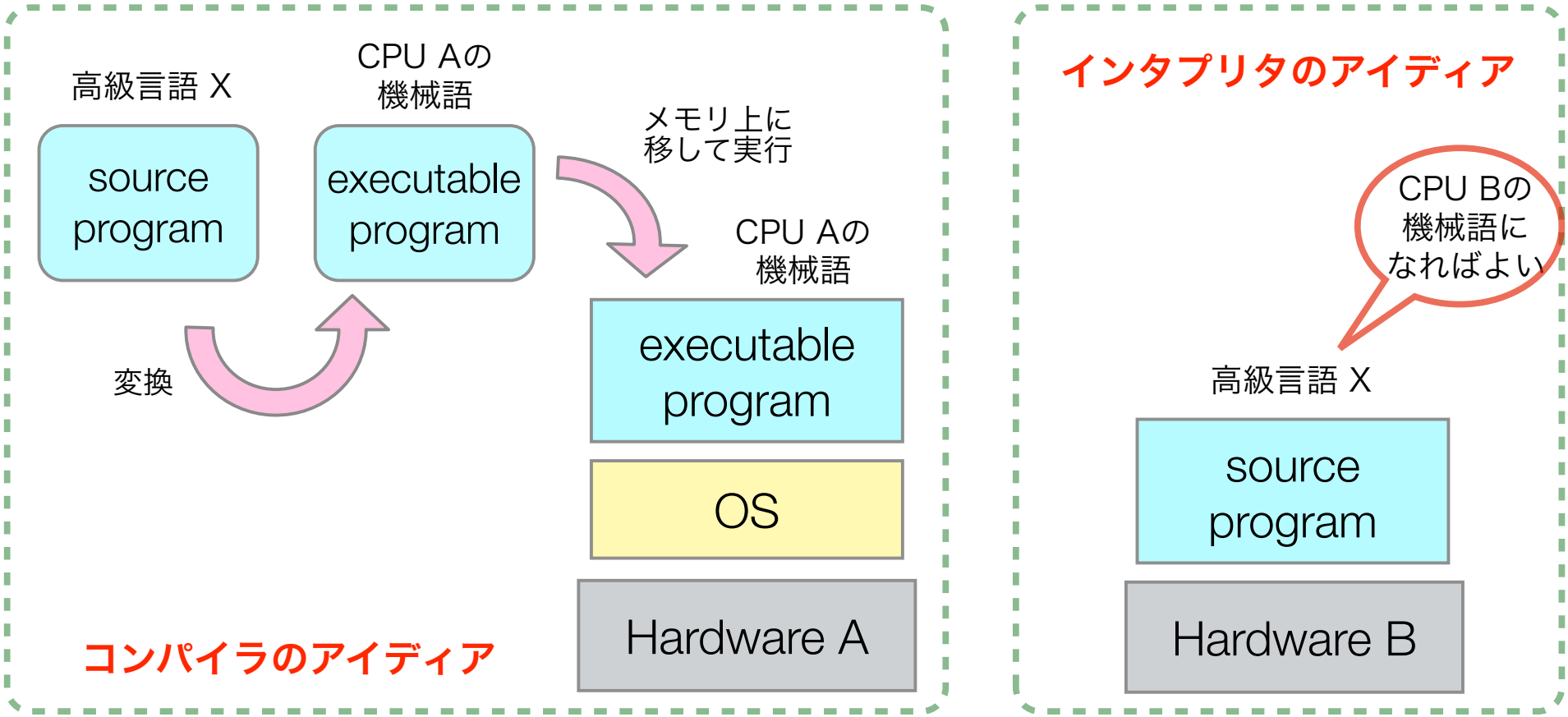


点線部分を PowerPC CPU の Macintosh システムと見なす

（厳密には Rosetta はコード変換実行システムであり、仮想マシンのシステムではない）

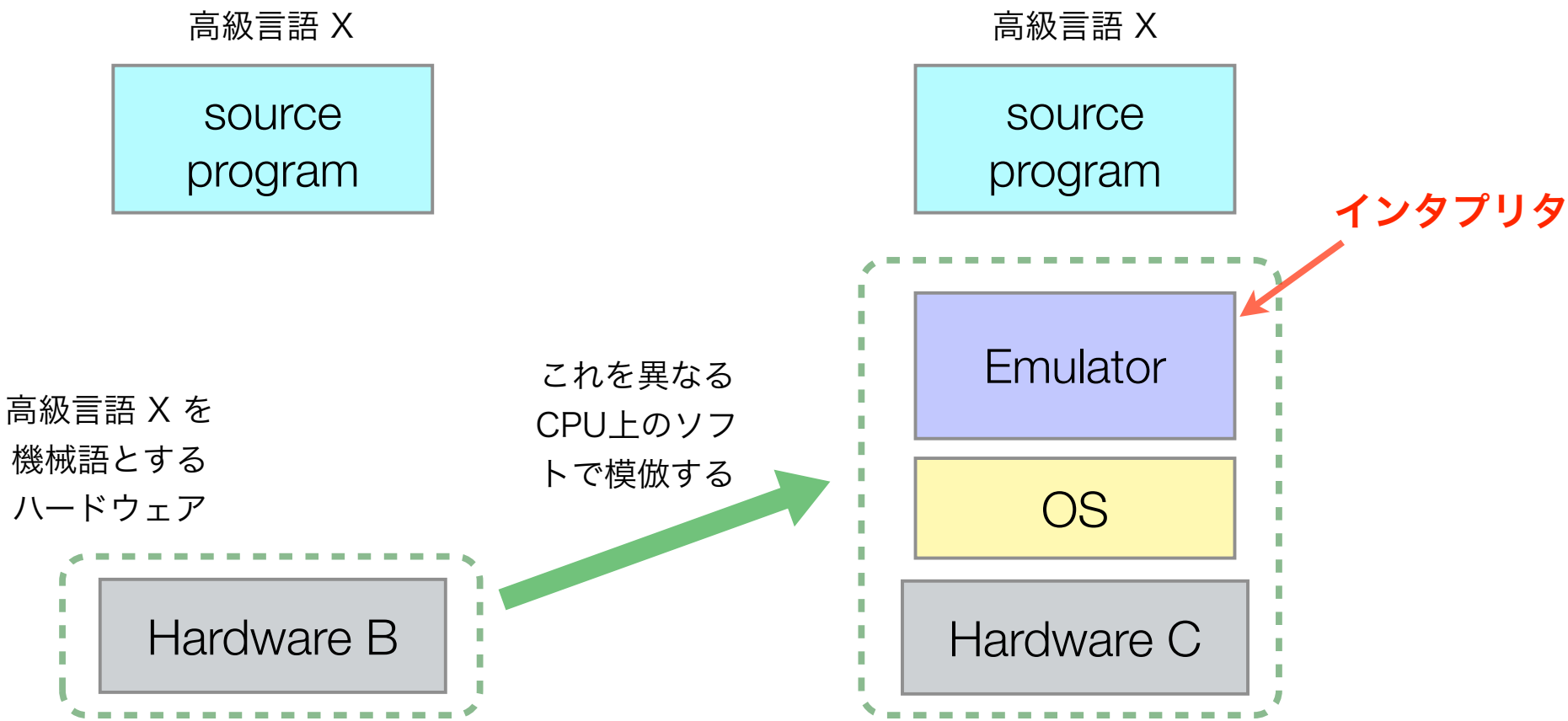
インタプリタ（再掲）

- その言語を直接実行できる CPU をソフトウェアで作る
- 「仮想的なコンピュータ」を考える

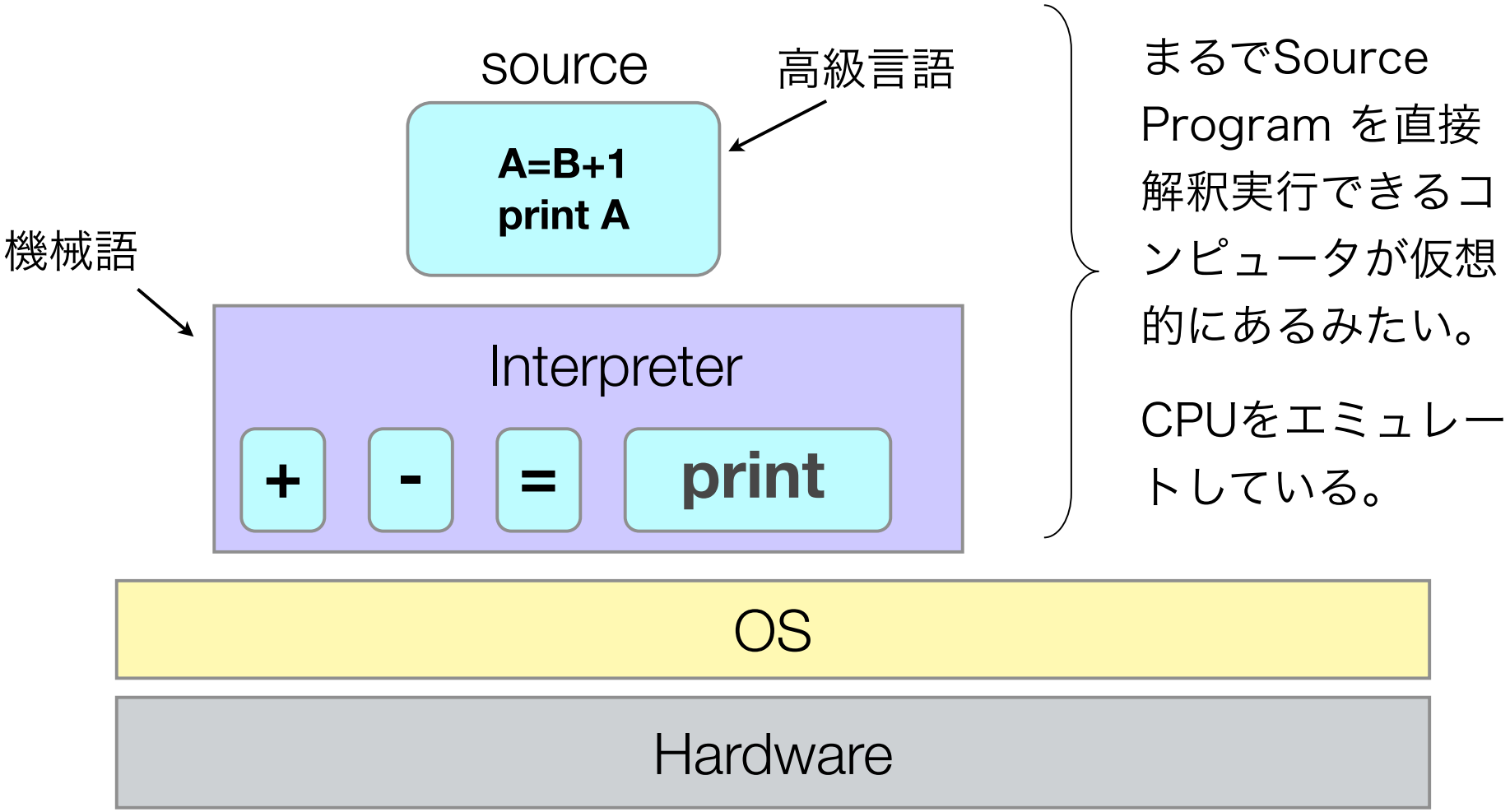


インタプリタ（再掲）

- その言語を直接実行できる CPU をソフトウェアで作る
- 「仮想的なコンピュータ」を考える



インタプリタ



まるでSource Program を直接解釈実行できるコンピュータが仮想的にあるみたい。

CPUをエミュレートしている。

変換はされていない点に注意

インタプリタ

- 実行したいプログラムを部分的に読み出し、対応する機械語コードを選択して実行

  読み出したときに文法的な解釈を行ってから対応する場合が多い

- Interpreter = 通訳、ではあるが変換はしていない

  逐語的に処理している点が重要（コンパイルは一括）

  まるでシミュレーション
  (simulate : 装う / emulation : 真似る)

- その言語を直接実行できる CPU をソフトウェアで仮想的に作ったようなもの

インタプリタ

- 部分的実行が可能

  一行ずつテストしながら開発するような作業が可能

- 実行速度が遅い

  真似るために必要な機械語は等価変換した機械語より処理量が多い

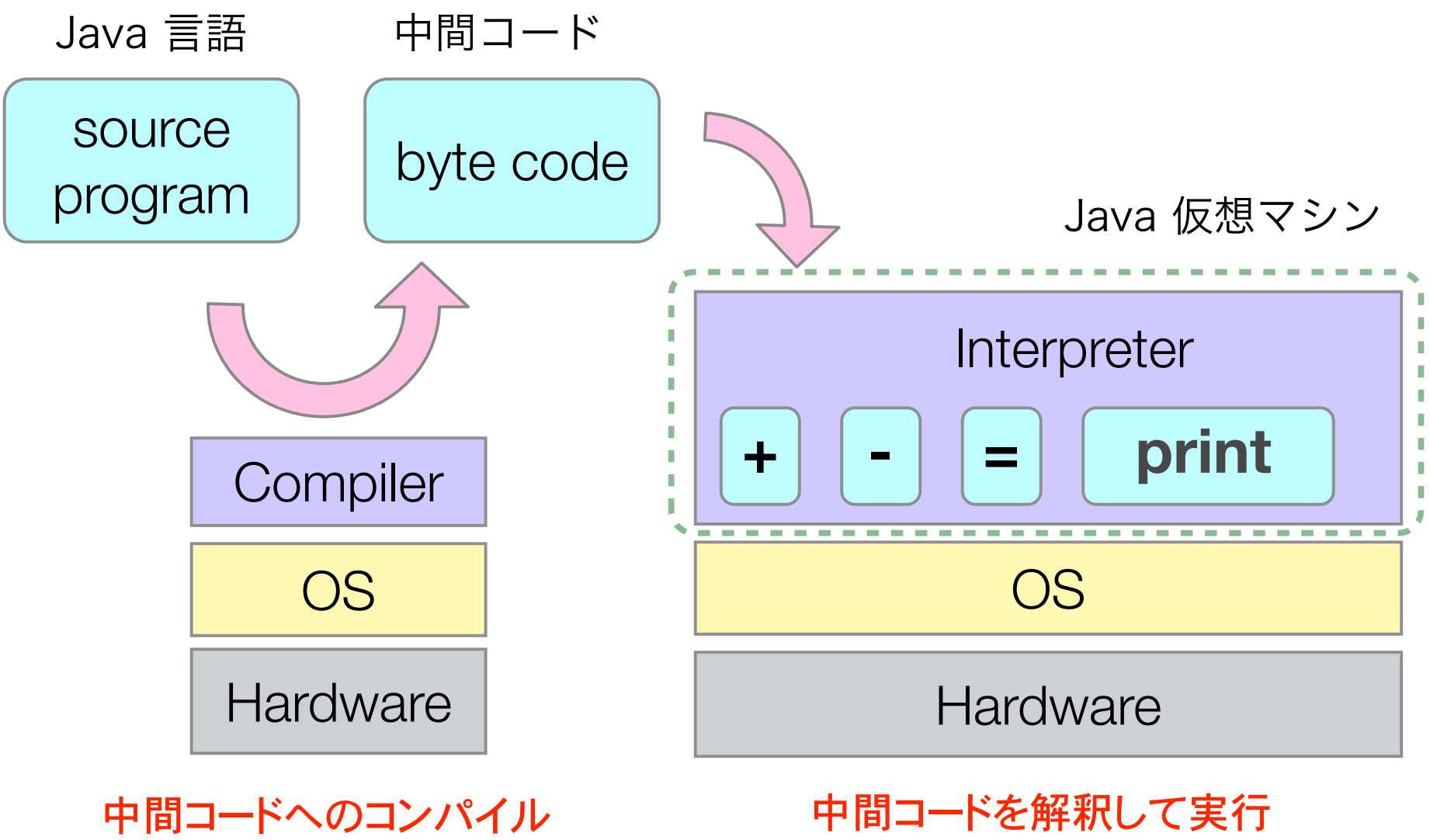
- CPU/機械語が異なる環境でも動作する

- Java / iアプリ

Java ：インタプリタによる言語の例

- Sun Microsystems が 1995 年に開発
  - C に似た文法
- まず中間プログラムにコンパイル
  - 実行速度を向上させるため
  - バイトコード (Byte Code) と呼ばれる
- インタプリタで実行
  - CPU やハードウェアの差を吸収して実行
  - Java VM (Virtual Machine 仮想計算機) の存在
  - iAPPLや Web で使われる

Java

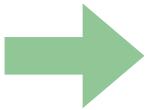

Java 言語
中間コード
source program
byte code
Java 仮想マシン
Interpreter
+
-
=
print
Compiler
OS
Hardware
OS
Hardware
中間コードへのコンパイル
中間コードを解釈して実行

Java ByteCode

Java言語

```
int a, b;
a=100;
b=200;
a=a+b;
```

高級言語



コンパイル

Java ByteCode

（ニモニック表記）

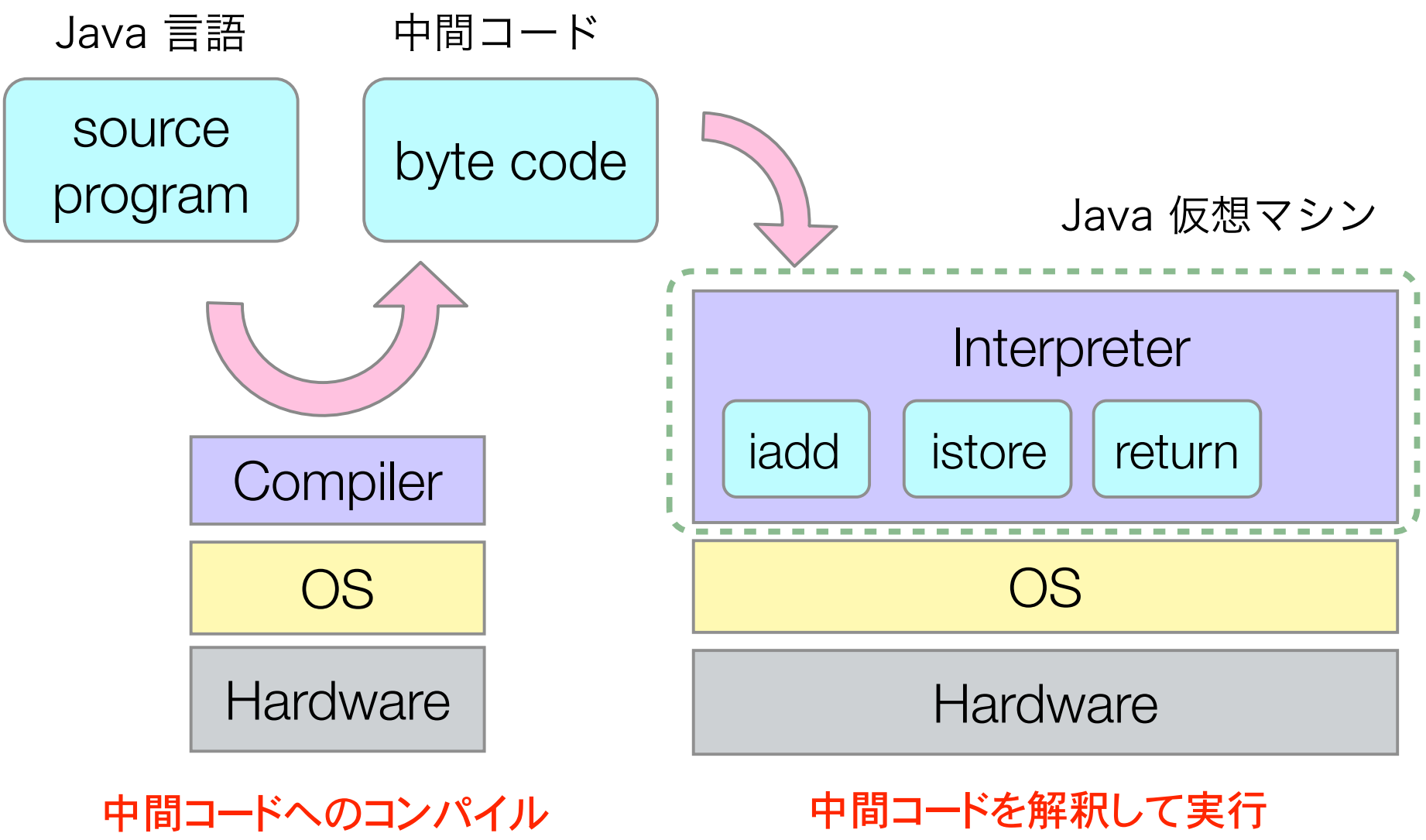
```
bipush 100
istore_1
sipush 200
istore_2
iload_1
iload_2
iadd
istore_1
return
```

（16進表記）

```
10 64          (10進表記では100)
3c
11 00 c8       (同200)
3d
1b
1c
60
3c
b1
```

中間コード（JavaVMでの機械語に相当）

Java


Java 言語
中間コード
source program
byte code
Java 仮想マシン
Interpreter
iadd
istore
return
Compiler
OS
Hardware
OS
Hardware
中間コードへのコンパイル
中間コードを解釈して実行

コンパイラとインタプリタ

- コンパイラ：一括変換して実行
- インタプリタ：逐次真似して実行
- 様々な方式がある

  Java のような組み合わせ
- Java 自身、VM 高速化のためにいろいろやっている

  Just In Time コンパイラ：バイトコード実行前に一括して機械語に変換、実行

  Hot Spot ：バイトコードの一部分（ループの中など）をコンパイルしながら実行
- 工夫の産物である

  無意味な分類にとらわれないように