プログラミング概論 & プログラミング基礎演習

(新潟大学　Gコード(教養)科目)

元木 達也 (講義, 木曜4限演習)
motoki@ie.niigata-u.ac.jp
棚橋 重仁 (水曜4限演習)
tanahashi@eng.niigata-u.ac.jp

0 ガイダンス

科目の概要:

- 次の様な人のための講義です。
  - ◇ データ処理の手順をコンピュータに指示することによって
    直接コンピュータを使ってみたい人。
  - ◇ それらの作業を通じて
    コンピュータというものを理解したい人。
- 「データ処理の手順をコンピュータに指示する」ための言語としては
  現在最も普及しているC言語を選ぶ。
- 例題を中心に説明する。

受講要件等:

内容を実際に体験するため

「プログラミング基礎演習」 (水曜４限または木曜４限)

も並行して履修することが望ましい。

受講に当っての留意事項:

- 全ての受講者がCプログラミングの実習が出来る環境にあることを想定して授業を進める。
- 数学的な内容の箇所もありますので、高校時代に「数学IA」しか履修していない人には厳しいかも知れません。
- 「プログラミング基礎演習」との同期をとるため、10月に補講を行うことがあります。
- 練習問題を時々出す予定。
  $\Longrightarrow$必ず自分で考えてやって下さい。
- 授業はほぼ講義ノートに沿って進める予定。
  $\Longrightarrow$できるだけ予習する。

科目のねらい:

- プログラミングを通じてコンピュータがどの様に動くのかを理解する。

学習の到達目標:

- 小規模な処理手順なら自分で設計できる。

教科書、参考書:

講義ノート(購入して下さい)に従って話を進める。

必要に応じて、参考書を読んで下さい。 例えば次の様な参考書がある。

- 蓑原隆「Cプログラミングの基礎」(2001年,サイエンス社, 1600円＋税)
- 皆本晃弥「やさしく学べるC言語入門 —基礎から数値計算入門まで—」(2004年,サイエンス社,2400円＋税)
- 鈴木正人「(情報学コアテキスト23) 実践Cプログラミング—基礎から設計/実装/テストまで—」(2008年,サイエンス社,1900円＋税)
- 柴田望洋「新版明解C言語 入門編」(2004年,ソフトバンククリエイティブ,2200円＋税)
- 柴田望洋「新版明解C言語 実践編」(2004年,ソフトバンククリエイティブ,2200円＋税)
- H.シルト「独習C 第4版」(2007年,翔泳社, 3200円＋税)

- 阿部圭一 (編)「(インターユニバーシティ) プログラミング」(1999年, オーム社,2300円＋税)
- 浦昭二&原田賢一 (編)「C入門」(1994年, 培風館, 2150円＋税)
- P.Prinz&U.Kirch-Prinz「Cデスクトップリファレンス」(2003年, オライリージャパン/オーム社,1200円＋税)
- B.W. カーニハン&D.M. リッチー「プログラミング言語C　第2版」(1989年, 共立出版, 2800円＋税)

授業予定:

［コンピュータ入門(I)］ —プログラミングの話をする前の準備 —

1 序論、コンピュータの一般的な構成、動作原理

1~ 2 コンピュータ内での情報の表現

［Cプログラミング入門］

3~ 4 プログラミング序論 —アルゴリズムの設計・記述—,
整数計算の簡単なプログラム例

4~ 6 処理の選択と繰り返し

6~ 8 実数データの扱い

8 数値計算 —方程式の解法, 数値積分—

9 配列 —「添字付きの名前」(例えばx_i)をCプログラムでどう表すか

10~ 11 関数の定義 —大規模な処理手順をC言語で表すための手法—

12~ 13 基本的なデータ型と構造体, 共用体

14 ファイル入出力

［コンピュータ入門(II)］ — プログラミングに関連した話 —

15 コンピュータのソフトウェア,
コンピュータ発展の歴史

試験

16 期末試験

成績評価の方法と基準:

上記目標の達成度を

- 期末試験 (80%),
- ほぼ毎週出す練習問題の出来具合等 (20%)

に基づいて評価し、60 点以上を合格とする。

但し、
授業の出席率が２／３未満の受講生については、
原則として単位を認めない。

1 序論

1–1 コンピュータの利用されている身近な例

コンピュータの利用例：

- 現象をシミュレーションによって解析,
  … (特に現実の実験が困難な現象 (e.g. 構造物の解析, 分子軌道の解析) に対して、シミュレーションは有効。スーパーコンピュータがよく使われる。)
- 組合せ最適化問題の最適解を探索,
  … (巡回セールスマン問題, スケジューリング問題など)
- 実験データの処理, … (実験結果の特徴を統計的にとらえるなど)
- 論文の執筆, … (ワードプロセッサ, LaTeX, お絵描きソフト)
- 画像処理, … (例えば、観測された画像からノイズを除去するなど)
- 新聞紙面の作成/編集, 電子出版, DTP(Desk Top Publishing),
- 自動車の設計, … (例えば、CAD(Compute Aided Design) を使って決まりきった部分はコンピュータに任せる。)
- 複数の産業用ロボットによる自動車の組立て,

… (各々の産業用ロボットにはその動作を制御する(マイクロ) コンピュータが置かれ、全てのロボットと自動車組立の流れを管理する計算機が中央に置かれる。)

ネットワーク経由での利用：

- 新潟大学学務情報システム,
- 新潟大学附属図書館所蔵の図書・雑誌の検索 (データベースの利用),
- 列車, 航空機の座席予約,
- 銀行の現金自動支払機,
- デパートや小売り店のPOS(Point Of Sales；販売時点情報管理),
- 住民基本台帳ネットワークシステム,

ネットワークと組み合わせて利用：

- インターネット, … (LAN同士を繋いで出来上がった世界最大規模のコンピュータネットワーク。 電子メール, WWW, ネットニュースなどのサービスがある。)
- イントラネット, … (インターネットと同様の仕組みを1つの会社内に閉じた形で構築したもの)

マイクロコンピュータチップ(5mm角位の大きさ)が組込まれた機器：

- 携帯電話,
- テレビゲーム機, 電卓, 時計, カメラ,
  ミシン, 炊飯器, 洗濯機, エレベータ,
  自動車(排ガス規制の克服,燃料消費効率の向上などのために)

1–2 コンピュータの普及に伴って起きる社会問題

- コンピュータ犯罪: 例えば、
  - ◇ コンピュータ・ウイルス,

補足：
自己増殖/伝染する様に作られた犯罪プログラムのこと。例えば、Nimda は 2001 年に新潟大学内を始め各地で被害をもたらし新聞等の記事にもなった。他にも、Brain(1986), Cascade(1987), Jerusalem(1987), Morris Worm(1988), Japanese Christmas (1989), Michelangelo(1991), Concept(1995), ..., sister(2002), Sasser(2004), Cabir(2004), 山田ウイルス (2005), polipos(2006), 山田オルタナティブ (2006), Gumblar(2009) 等があったらしい。
(http://ja.wikipedia.org/wiki/コンピュータウイルス)

  - ◇ コンピュータへの不正アクセス,

例えば、
1985 年、筑波の高エネルギー研究所の計算機に西ドイツからの不法侵入があり、大事なプログラムが壊された。

◇ コンピュータの不正使用,

例えば、

— 金融機関の内部職員が虚偽のデータを計算機に入力して、架空名義の預金口座に入金があったように見せかける。

— トロイの木馬：プログラムの中に内密の処理手順を挿入しておき、(プログラムの本来の目的を果たさせながら) 無許可の機能も実行させる。

— サラミ・テクニック：トロイの木馬的な方法により、多くの財産・資源から少しずつ (e.g. 利息計算の際の端数の処理に細工して) 目立たない様に盗みとる。

◇ 有料ソフトウェアを無断でコピーして販売・使用,

◇ 顧客データをコピーして他社に売却,

◇ インターネット上で個人中傷,

◇ インターネット詐欺,

例えばオンラインショッピングによる代金, 商品の横領

◇ インターネットを利用したネズミ講, マルチ商法

◇ 非合法な物品 (薬物, 危険物など) や情報の販売

◇ キャッシュカードを不正に作出し、それを使って現金を引き出す。

- 新しい職業病: 例えば、
  - ◇ VDT(Video Display Terminal) 作業による視力障害,
  - ◇ コンピュータ依存症 (対人恐怖，働き過ぎ),
  - ◇ コンピュータ拒否症,
  - ◇ IC工場での極度のストレス
- 計算機システムの故障による混乱: 例えば、JRの列車の座席予約を管理する計算機が地震などで停止すると、(一時的にではあるが) 日本中が混乱する。また、管理プログラムの小さな誤りのために大事なファイルが消されたりすると、しばらく混乱する。
- 雇用の問題: 「計算機を導入して業務を合理化すると人が要らなくなるのではないか？」という不安。
- プライバシーの侵害: 個人のデータを（無断で）集めることによって起こる問題。例えば、ある犯罪の記録が事件に全く無関係な人のデータとして誤って入力され、この記録が信用調査録として広く行き渡ったとしたら、この人は知らぬ間に不利益を被ることになる。しかも、この人は誤ったデータを見ることも訂正することもできない。

- ソフトウェアの著作権: ソフトウェアやホームページ上の文書や画像は従来の著作物とは異なる性質を持つ。これにどういう風に知的所有権を認めるか?
- インターネット中毒:

コンピュータについての正しい理解は現代では不可欠。

⇨ 正しい理解のための基礎を与えるのが
この講義の目的
(プログラミングを通じて
コンピュータの動作を理解する。)

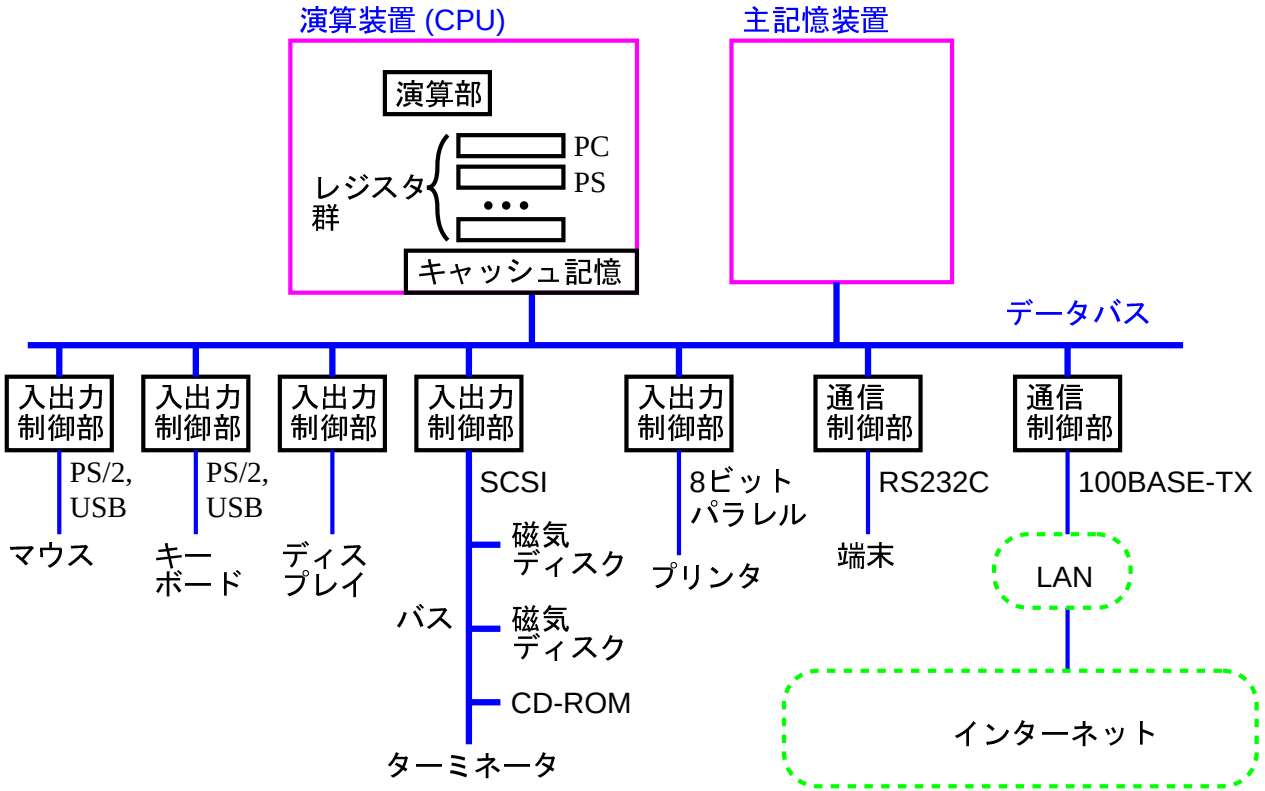
2 コンピュータの構成、動作原理

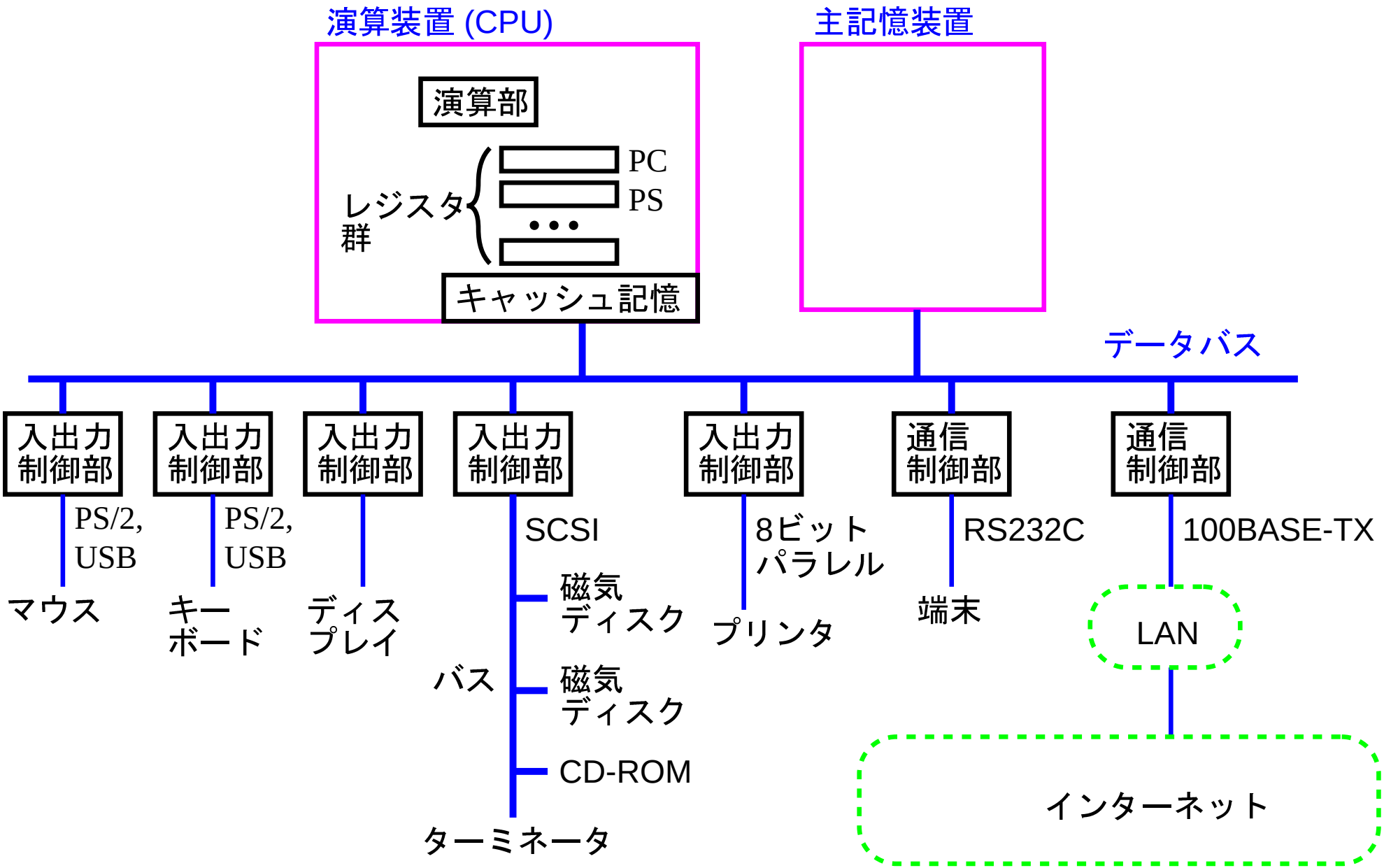
2–1 コンピュータのハードウェア構成

ハードウェアは、主に

- 演算装置 (CPU, プロセッサ),
- 主記憶装置 (メモリ),
- 入出力制御部 (または通信制御部),
- バス,
- 通信路,
- 周辺装置

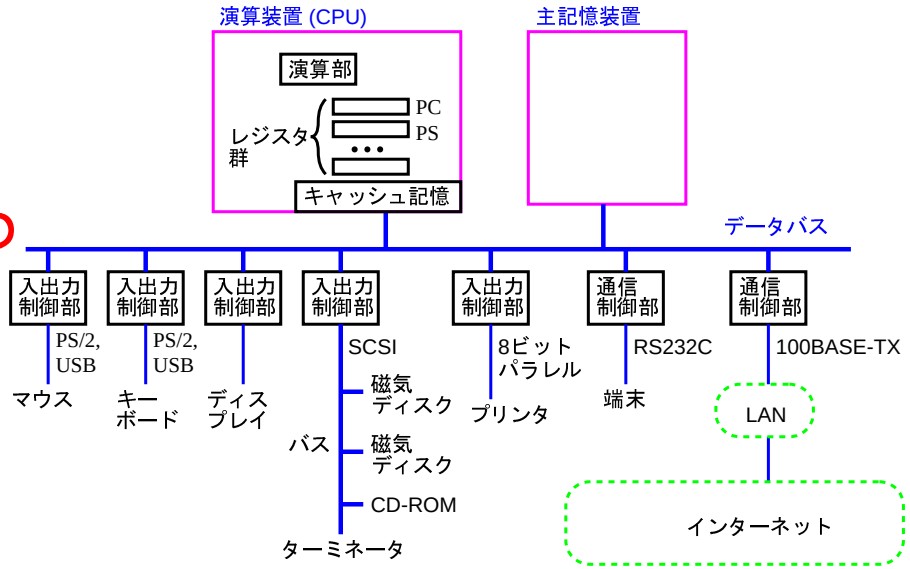
から成る。




演算装置 (CPU)
主記憶装置
演算部
PC
PS
レジスタ群
キャッシュ記憶
データバス
入出力制御部
入出力制御部
入出力制御部
入出力制御部
入出力制御部
通信制御部
通信制御部
PS/2, USB
PS/2, USB
SCSI
8ビット パラレル
RS232C
100BASE-TX
マウス
キーボード
ディスプレイ
磁気ディスク
磁気ディスク
CD-ROM
バス
ターミネータ
プリンタ
端末
LAN
インターネット

演算装置 (CPU, プロセッサ) :

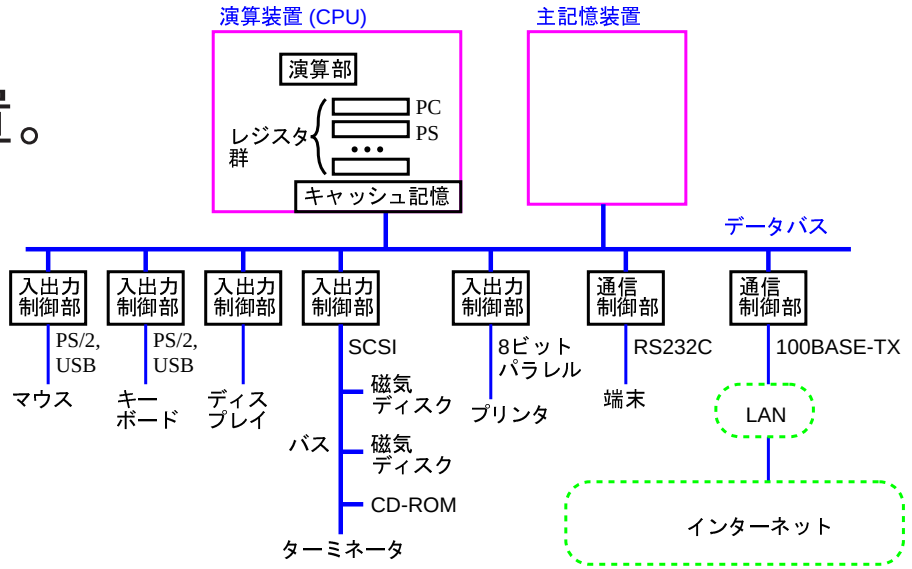
計算機システムの処理を司る部分。



- 主記憶装置から機械語命令を1つずつ順番に読み込み実行していく。
  ($\Longrightarrow$ 2.2節プログラム内蔵方式)
- 必要に応じて、データを主記憶装置から読み込んだり実行結果を主記憶装置に格納したりする。
- 主記憶装置との間のデータ転送はバスを介して行う。
- 演算・データ処理のために多くのレジスタを持っている。
  (1) 演算用レジスタ : 演算対象のデータ等を入れる。
  (2) 制御用レジスタ : プロセッサの動作を制御するためのもの。
  - プログラムカウンタ ... 次の実行する命令のアドレスを記憶
  - プロセッサ状態レジスタ ... プロセッサの状態を保持

主記憶装置(メモリ):

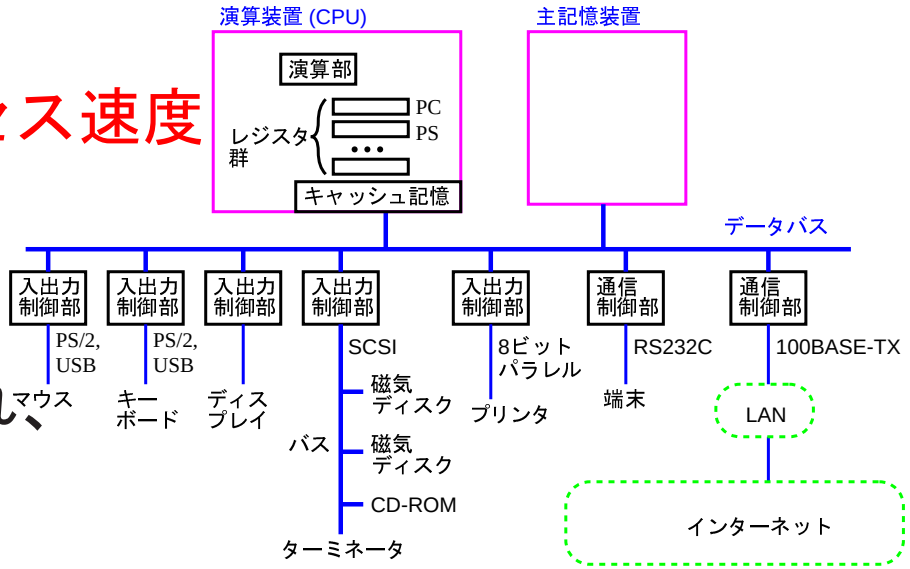
CPUから直接アクセスできる記憶装置。



- 使用中のデータだけでなく
  実行中のプログラム(の断片)も
  必ずこの中に入れておく。
- データの記憶場所を識別するために
  記憶領域には番地(address)が付けられ、
  CPUはこの番地を指定することにより
  主記憶内のデータの読み書きを行う。
- プロセッサやバスとの関係にも依存するが、
  連続する複数番地のデータを一度にまとめて読み書きできる。
- メモリの多くは揮発性である。

キャッシュ記憶：

CPUから主記憶内のデータへのアクセス速度を見かけ上高速化するための高速記憶。

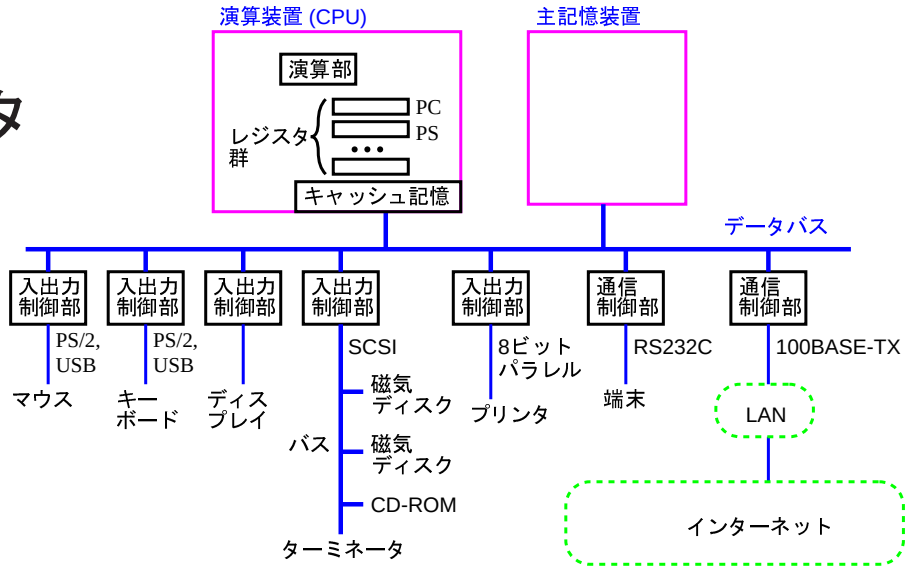


- アクセスされた主記憶の情報はキャッシュ記憶内に一時的に格納され、近い将来再びアクセスされた時はキャッシュ記憶から取り出される。
- メモリへのアクセス回数を抑えるため、キャッシュ記憶に無いデータをメモリから読み込む場合には、必要とするデータだけでなく近くのデータも一緒に読み込んで複製しておく。
- プログラムの実行が局所参照性を持っているので、キャッシュ記憶を用いることによって主記憶内のデータへの実際上のアクセススピードを上げることが可能になる。

入出力制御部：

周辺装置、端末装置、他のコンピュータとの間の入出力を行う。

- 相手の装置が能動的である場合には通信制御部と呼ぶ。



バス：

プロセッサ、メモリ、入出力制御部を結んで、それらの間のデータ授受を行うためのもの。

- 8ビット以上のデータを並列に転送する。

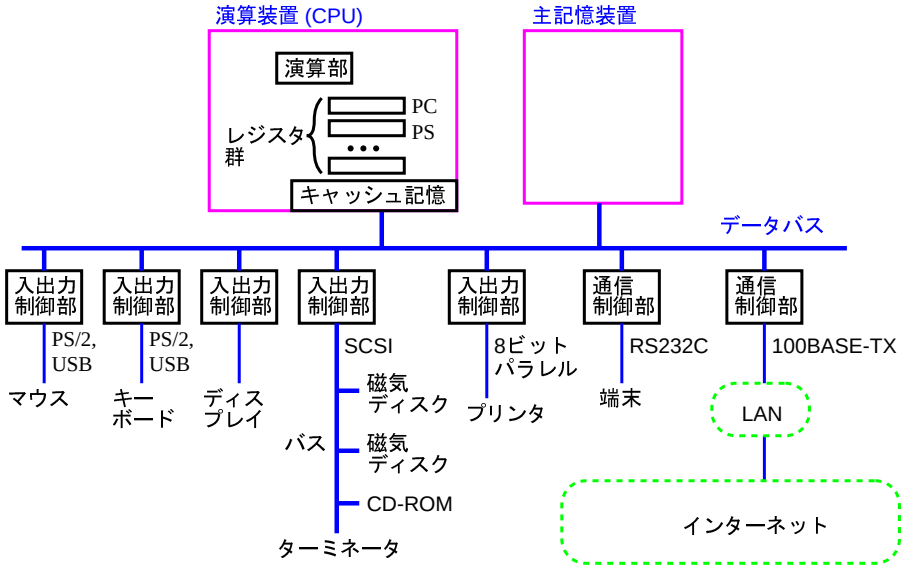
通信路：

入出力装置との接続には様々な通信路を用いる。　次のような接続規格がある。[参考文献：小高知宏「計算機システム」(森北出版, 1999)]

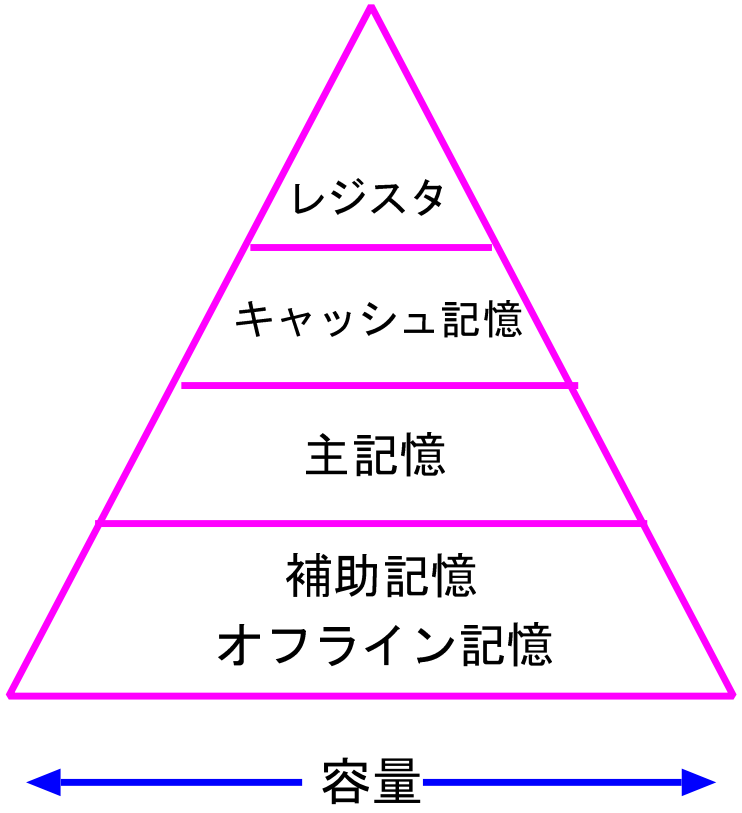
名称	接続装置	特徴
RS-232C	モデム, 端末装置, プリンタ	低速のシリアルインターフェース
8ビットパラレル (セントロニクス)	プリンタ	低速のパラレルインターフェース
SCSI	磁気ディスク, CD-ROM, MO, イメージスキャナ	高速 (5~ 80Mbytes/s) のパラレルインターフェース
USB	キーボード, マウス, プリンタ, モデム, ディジタルカメラ	高速 (最大 12Mbits/s) の汎用インターフェース。キーボード, マウス, RS-232C などの低速な入出力装置が抱えている様々な問題を解決するために開発された。複数の機器をバス接続できる。
IEEE1394 (Fire Wire)	マルチメディア入出力装置	高速 (最大 400Mbits/s) の汎用インターフェース
ファイバーチャネル	高速ディスク装置	高速 (1Gbits/s) の汎用インターフェース
100BASE-TX	LAN	100Mbits/s, ツイストペア線
1000BASE-T	LAN	1Gbits/s, ツイストペア線

周辺装置：

計算機本体の周辺に置かれる装置を総称して言う。例えば、
磁気ディスク、プリンタなど。



記憶装置の階層：



高価、高速

低速、安価

2–2 コンピュータの動作原理

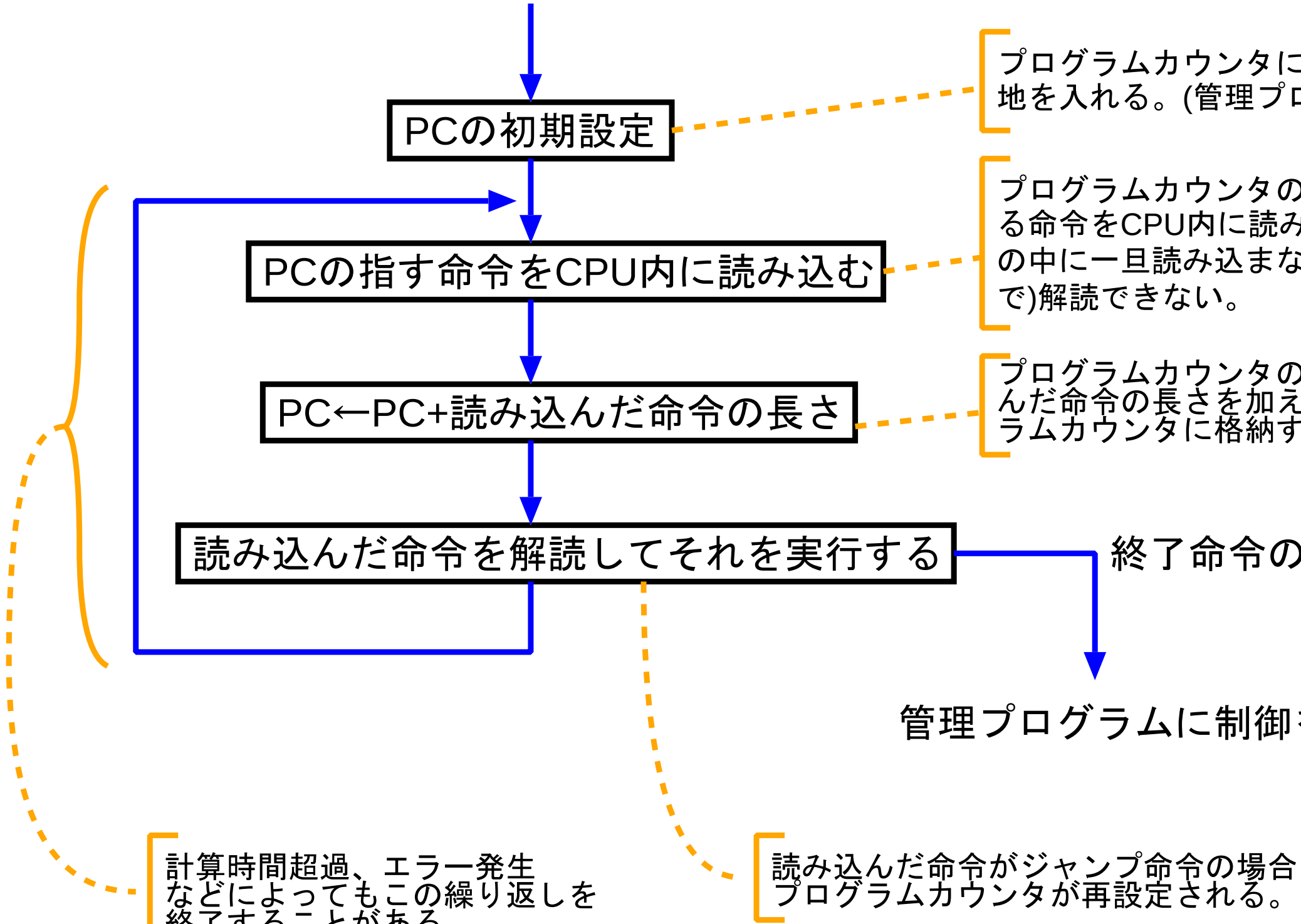
現在の普通の計算機においては、

- プログラム (i.e. 処理手順) をデータと同様に記憶装置内に格納し、(プログラムをデータとして加工できる)
- プログラムを構成する機械語命令を逐次的に読み出しては実行していくことにより自動的に動作させる、

いわゆるプログラム内蔵方式 (stored program system, またはノイマン型) が採用されている。

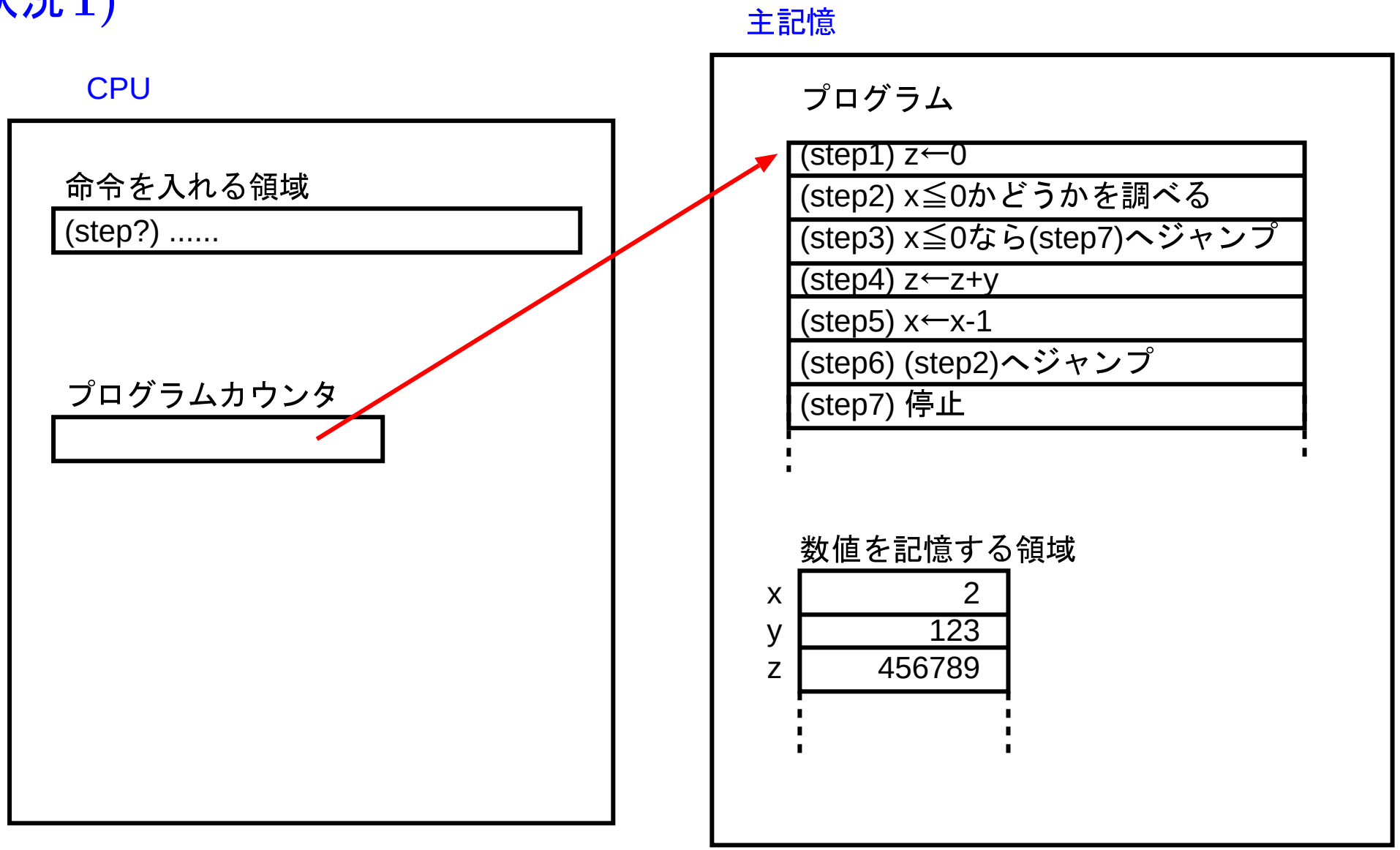
プログラム内蔵方式の計算機では 、

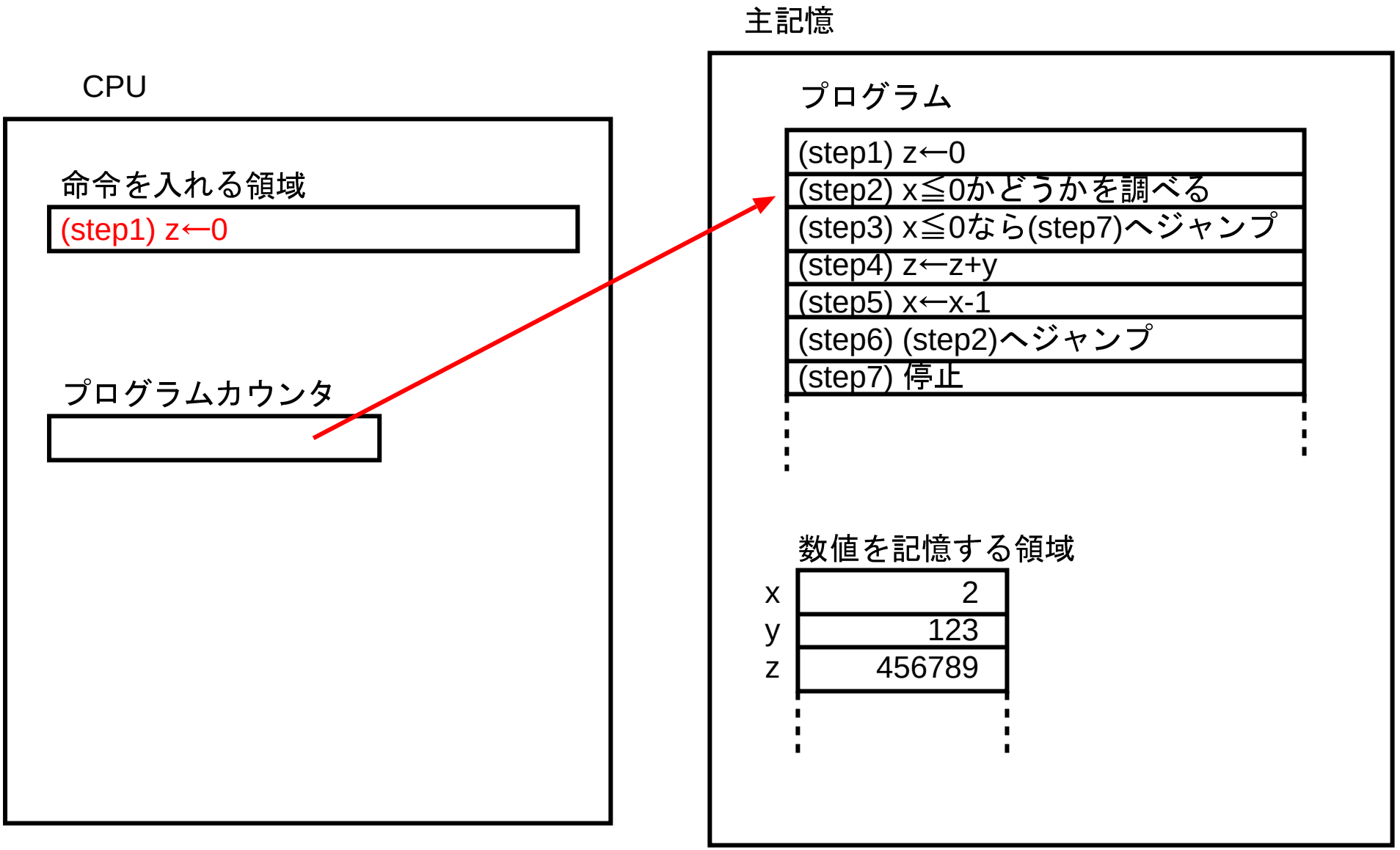
- プログラムカウンタと呼ばれる
  次に実行する命令の入った (主記憶上の) 番地
  を常に保持するレジスタ記憶を用意する。
- CPU はプログラムカウンタを用いて
  プログラム中の命令を逐次実行する。


PCの初期設定
PCの指す命令をCPU内に読み込む
PC←PC+読み込んだ命令の長さ
読み込んだ命令を解読してそれを実行する
プログラムカウンタに
地を入れる。(管理プロ
プログラムカウンタの
る命令をCPU内に読み
の中に一旦読み込まな
で)解読できない。
プログラムカウンタの
んだ命令の長さを加え
ラムカウンタに格納す
終了命令の
管理プログラムに制御
計算時間超過、エラー発生
などによってもこの繰り返しを
読み込んだ命令がジャンプ命令の場合
プログラムカウンタが再設定される。

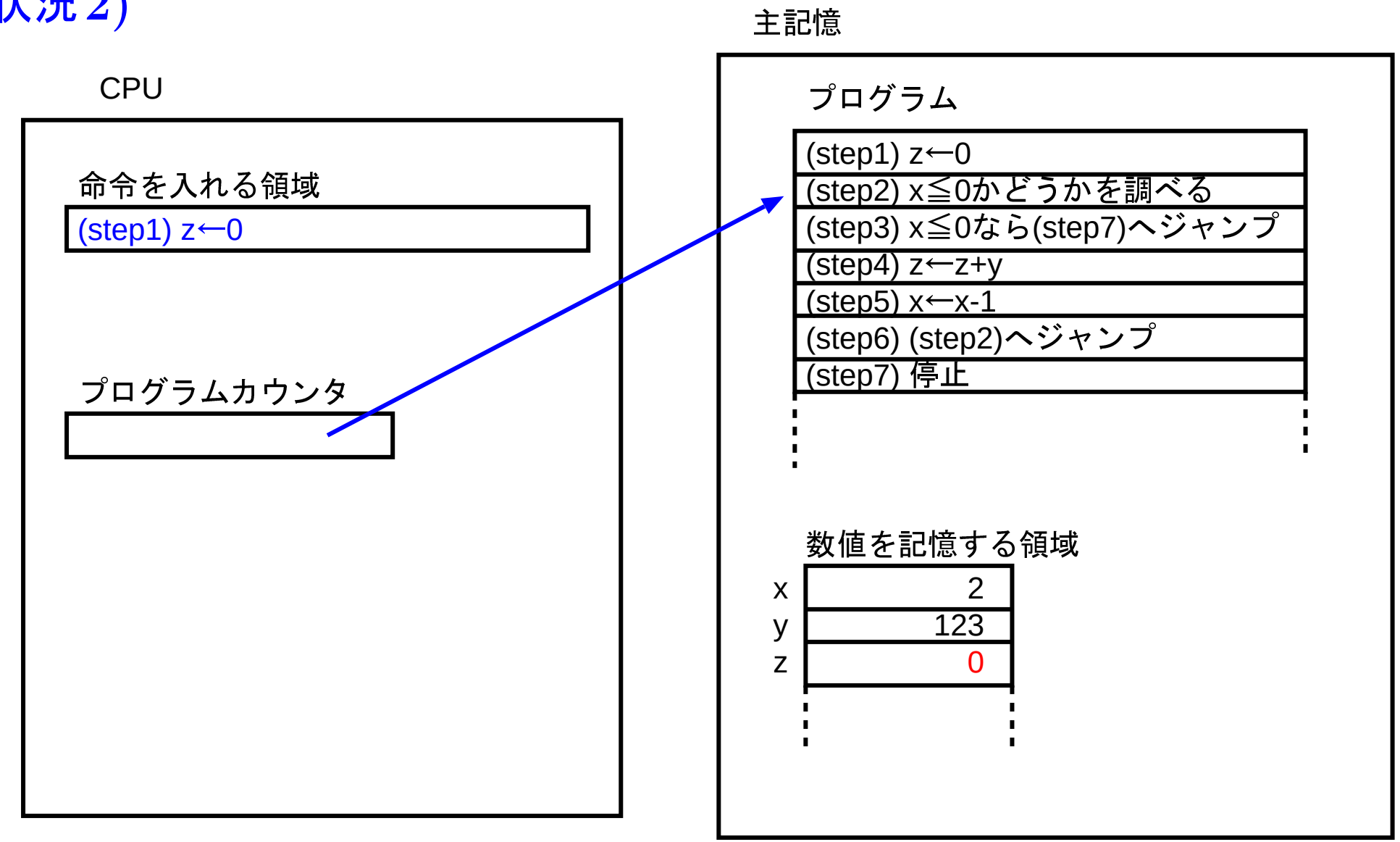
例2. 1 (プログラム内蔵方式計算機の動作)

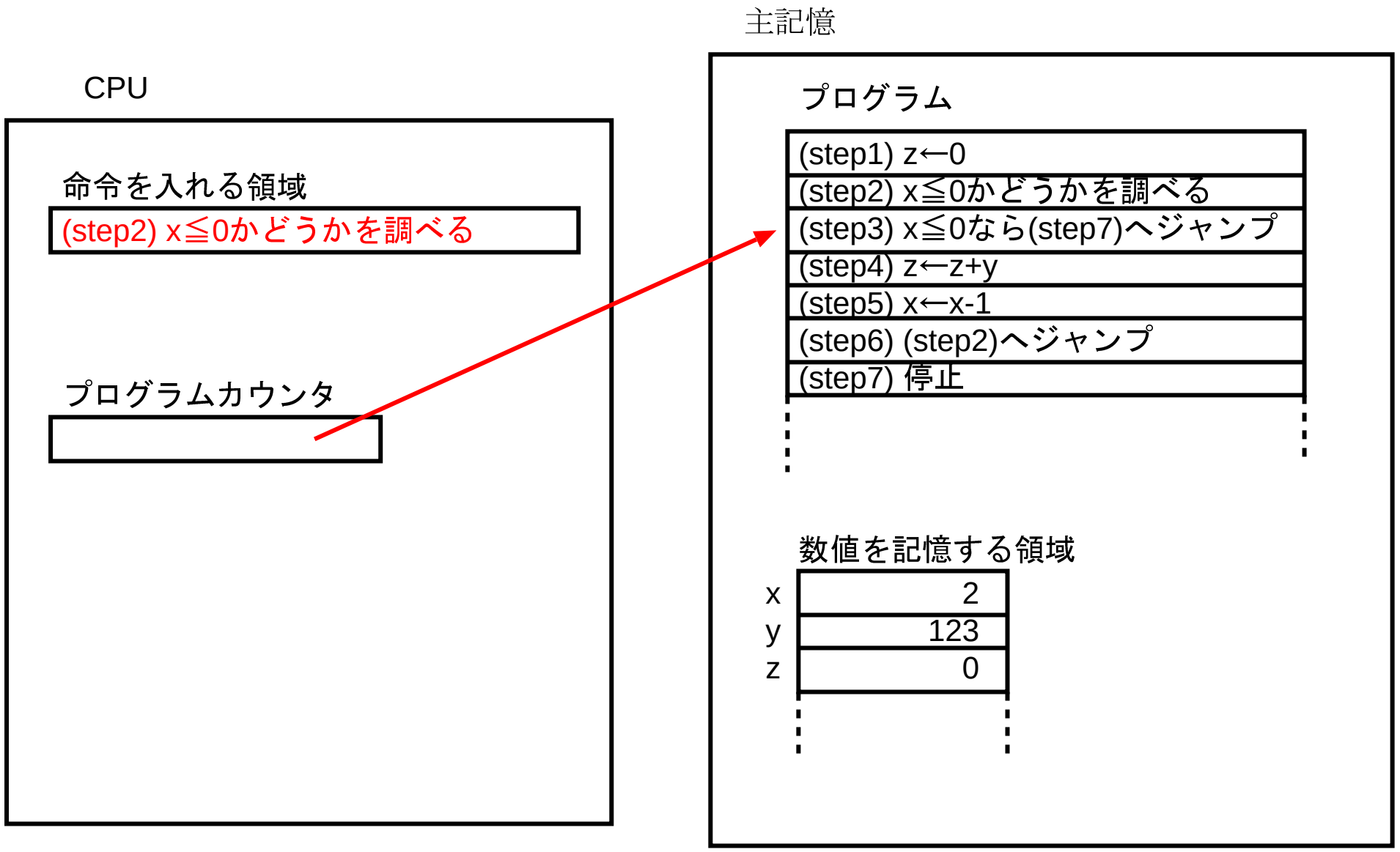
(状況1)



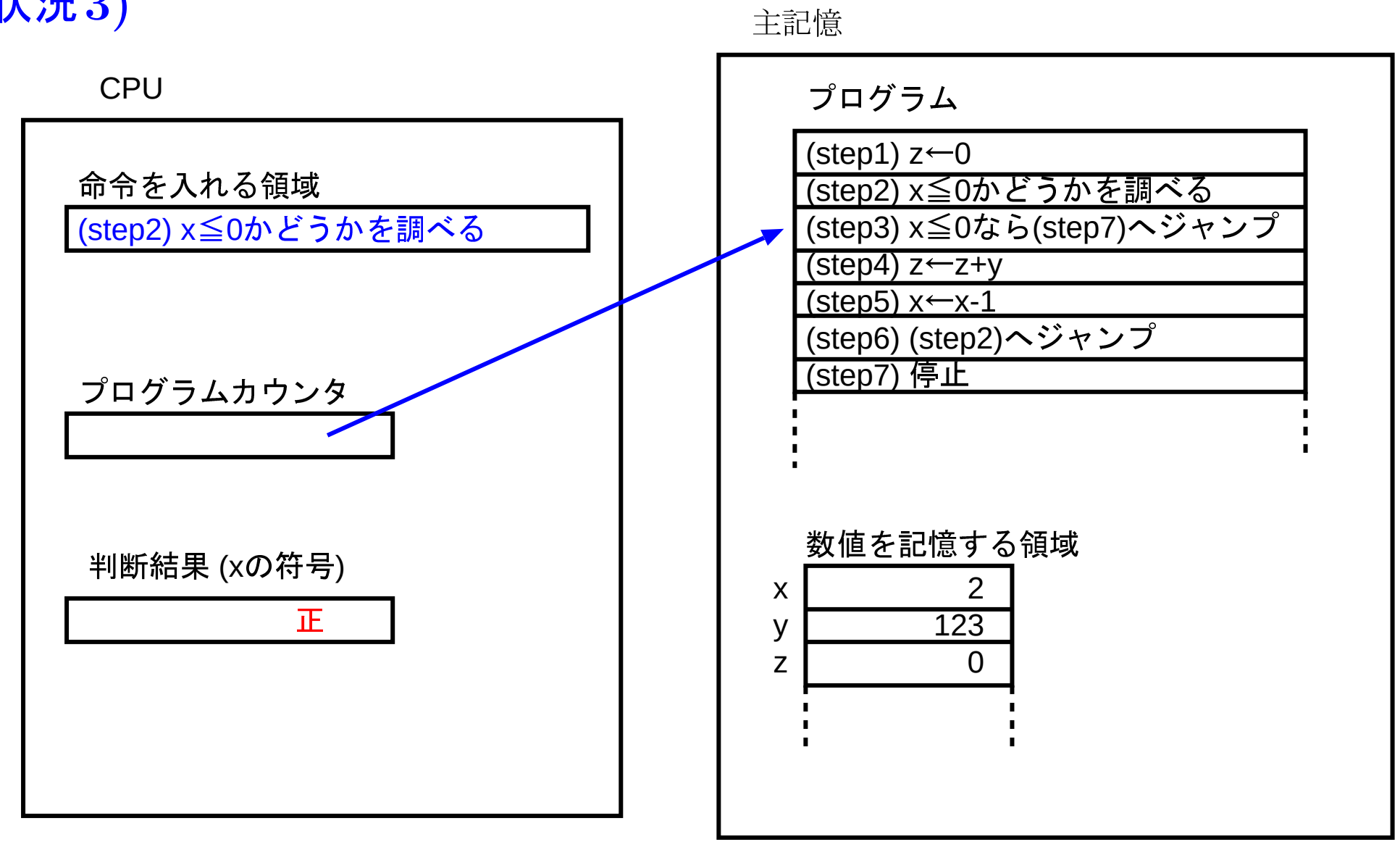

主記憶
CPU
プログラム
命令を入れる領域
(step1) z←0
(step2) x≦0かどうかを調べる
(step3) x≦0なら(step7)へジャンプ
(step4) z←z+y
(step5) x←x-1
(step6) (step2)へジャンプ
(step7) 停止
プログラムカウンタ
数値を記憶する領域
x
y
z
2
123
456789

(状況2)

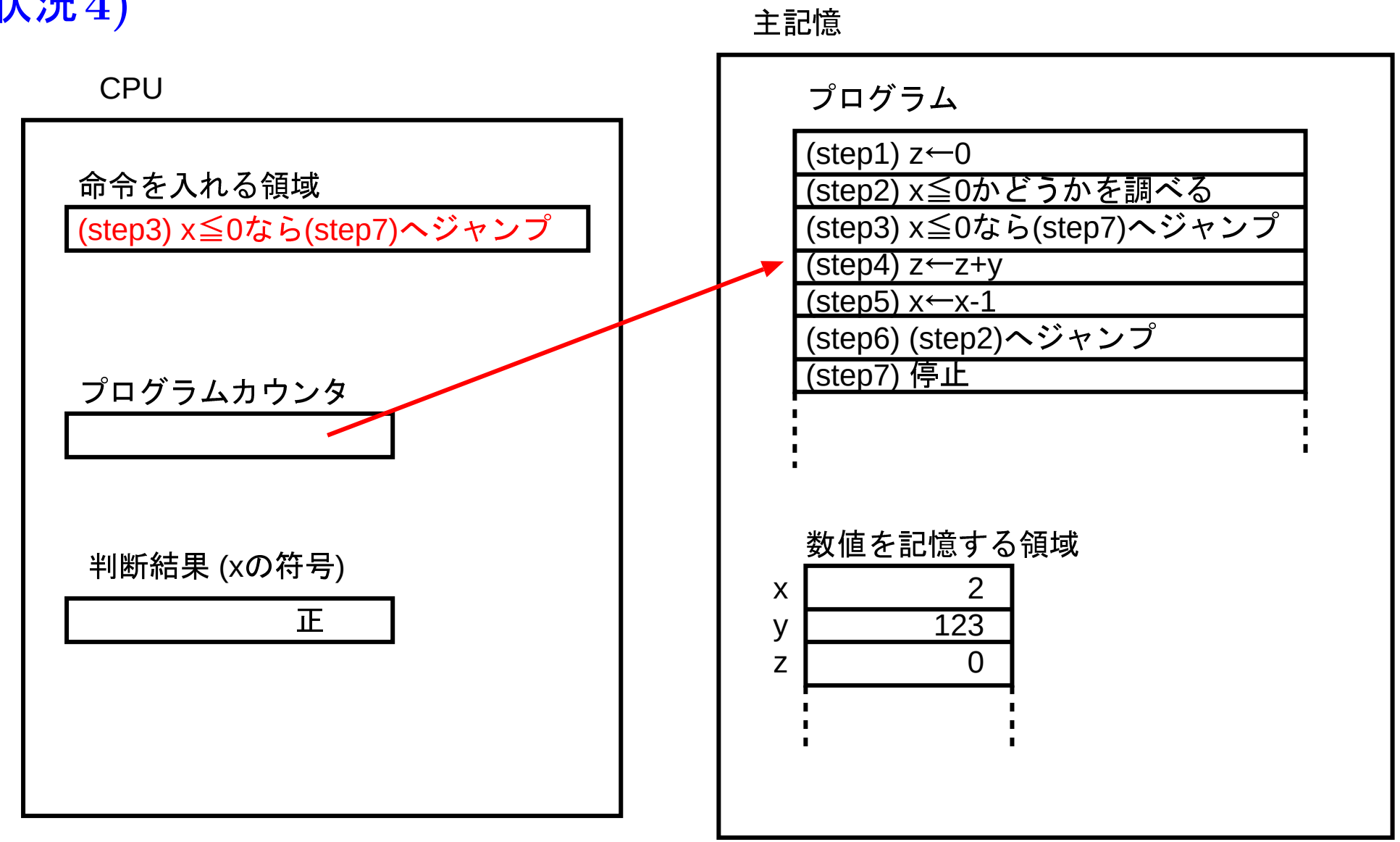

主記憶
CPU
プログラム
命令を入れる領域
(step1) z←0
(step2) x≦0かどうかを調べる
(step3) x≦0なら(step7)へジャンプ
(step4) z←z+y
(step5) x←x-1
(step6) (step2)へジャンプ
(step7) 停止
プログラムカウンタ
数値を記憶する領域
x
y
z
2
123
0

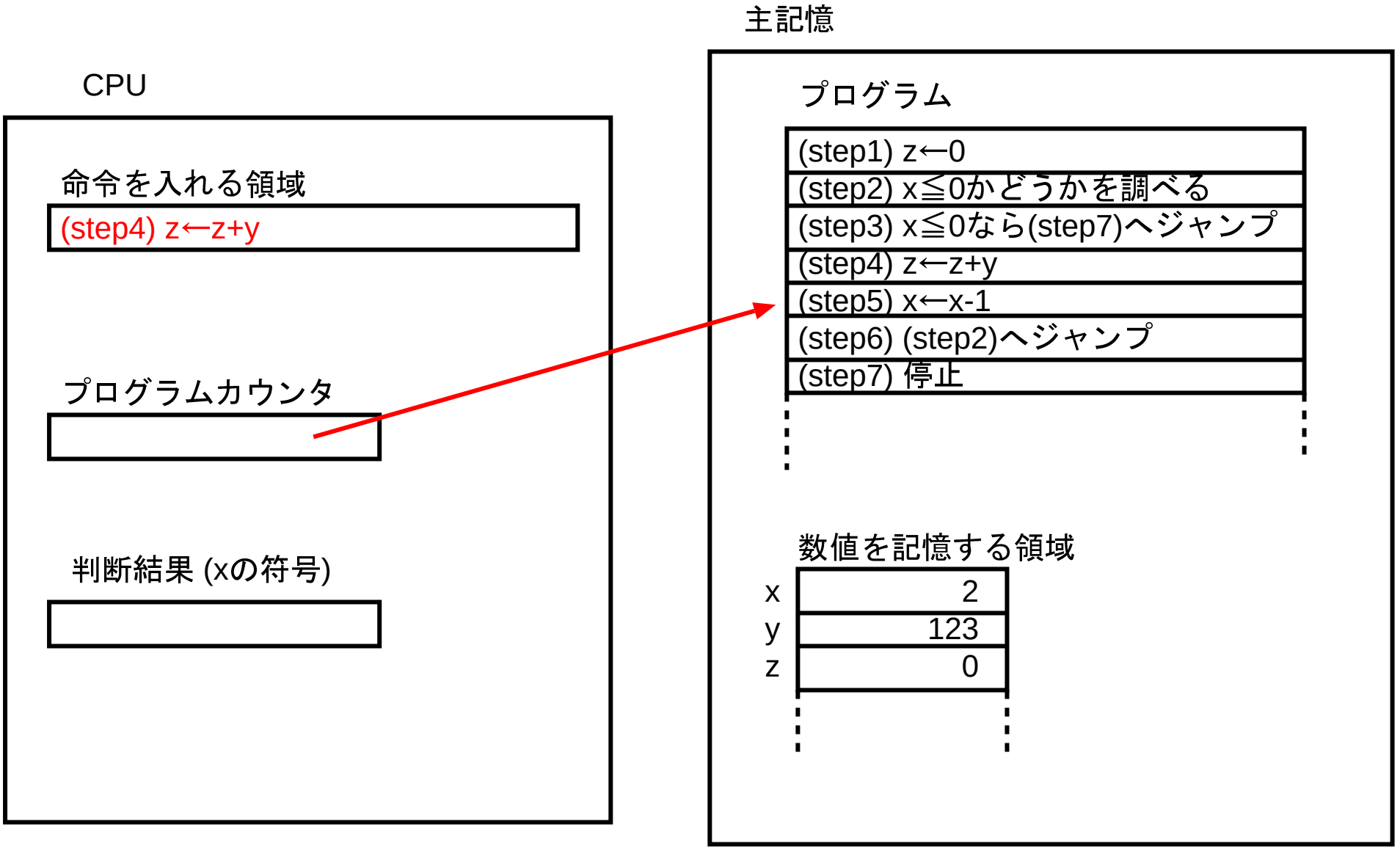

主記憶
CPU
命令を入れる領域
(step2) x≦0かどうかを調べる
プログラムカウンタ
プログラム
(step1) z←0
(step2) x≦0かどうかを調べる
(step3) x≦0なら(step7)へジャンプ
(step4) z←z+y
(step5) x←x-1
(step6) (step2)へジャンプ
(step7) 停止
数値を記憶する領域
x 2
y 123
z 0

(状況3)

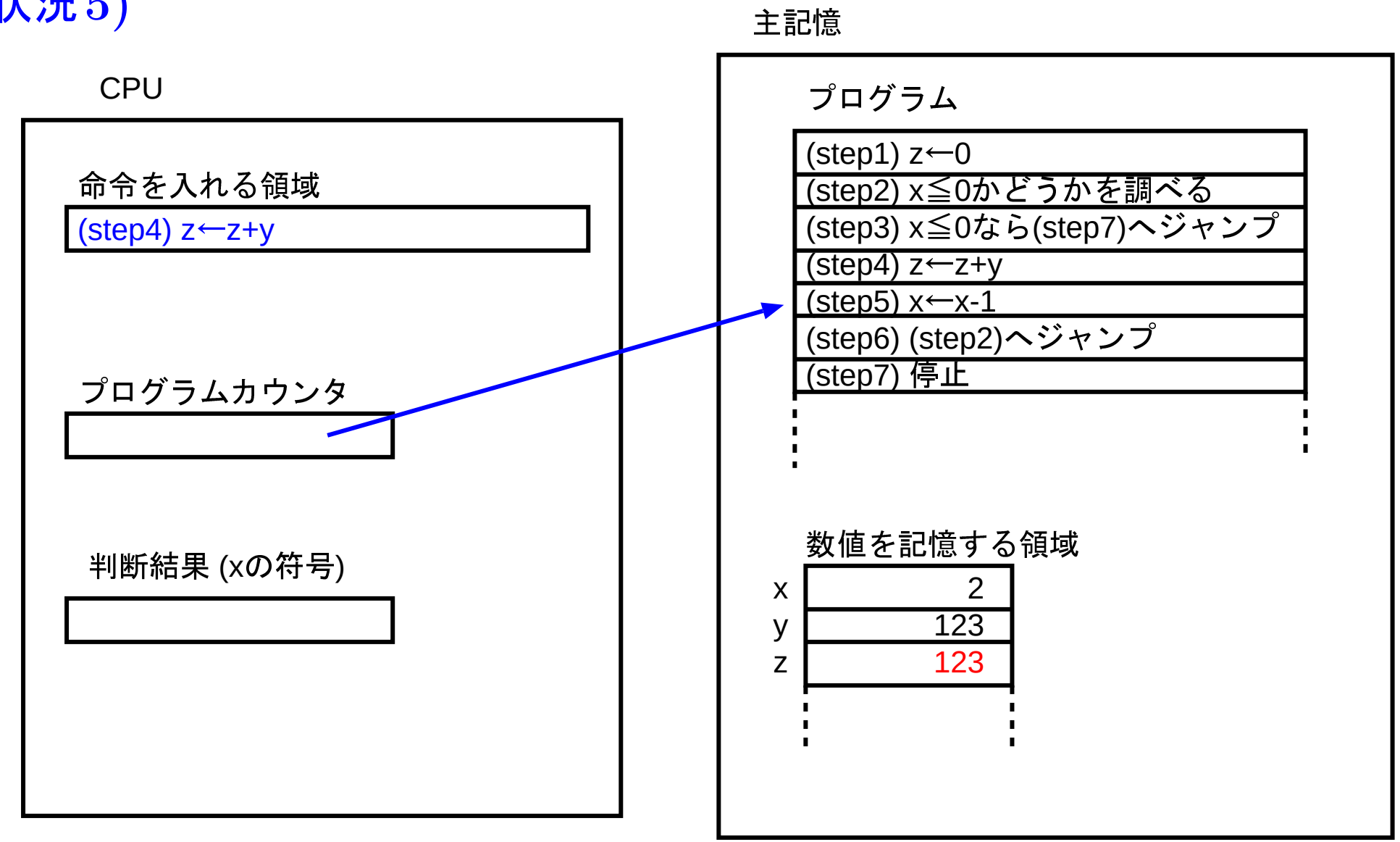

主記憶
CPU
命令を入れる領域
(step2) x≦0かどうかを調べる
プログラムカウンタ
判断結果 (xの符号)
正
プログラム
(step1) z←0
(step2) x≦0かどうかを調べる
(step3) x≦0なら(step7)へジャンプ
(step4) z←z+y
(step5) x←x-1
(step6) (step2)へジャンプ
(step7) 停止
数値を記憶する領域
x 2
y 123
z 0

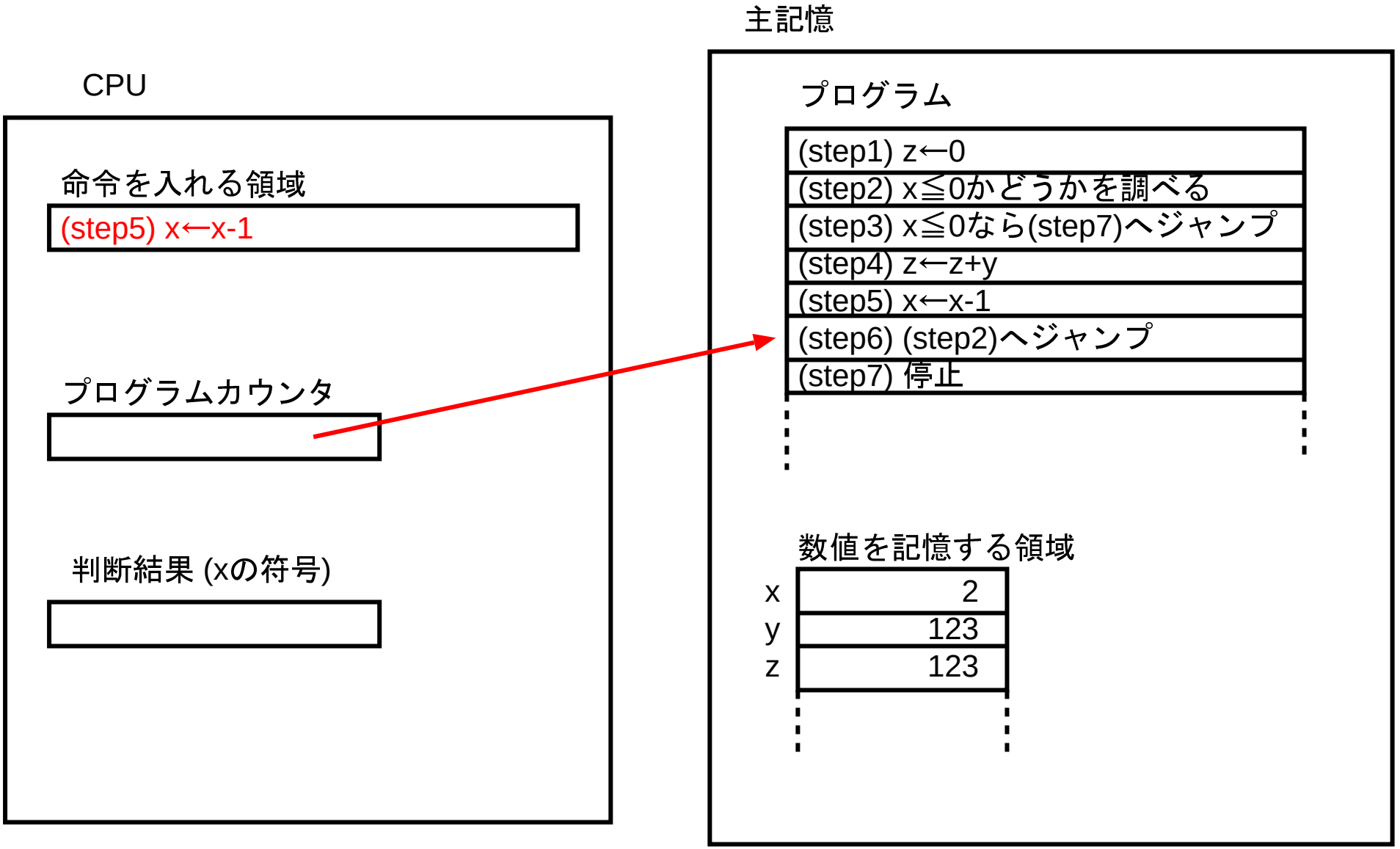
(状況4)


主記憶
CPU
プログラム
(step1) z←0
(step2) x≦0かどうかを調べる
(step3) x≦0なら(step7)へジャンプ
(step4) z←z+y
(step5) x←x-1
(step6) (step2)へジャンプ
(step7) 停止
命令を入れる領域
(step3) x≦0なら(step7)へジャンプ
プログラムカウンタ
判断結果 (xの符号)
正
数値を記憶する領域
x 2
y 123
z 0

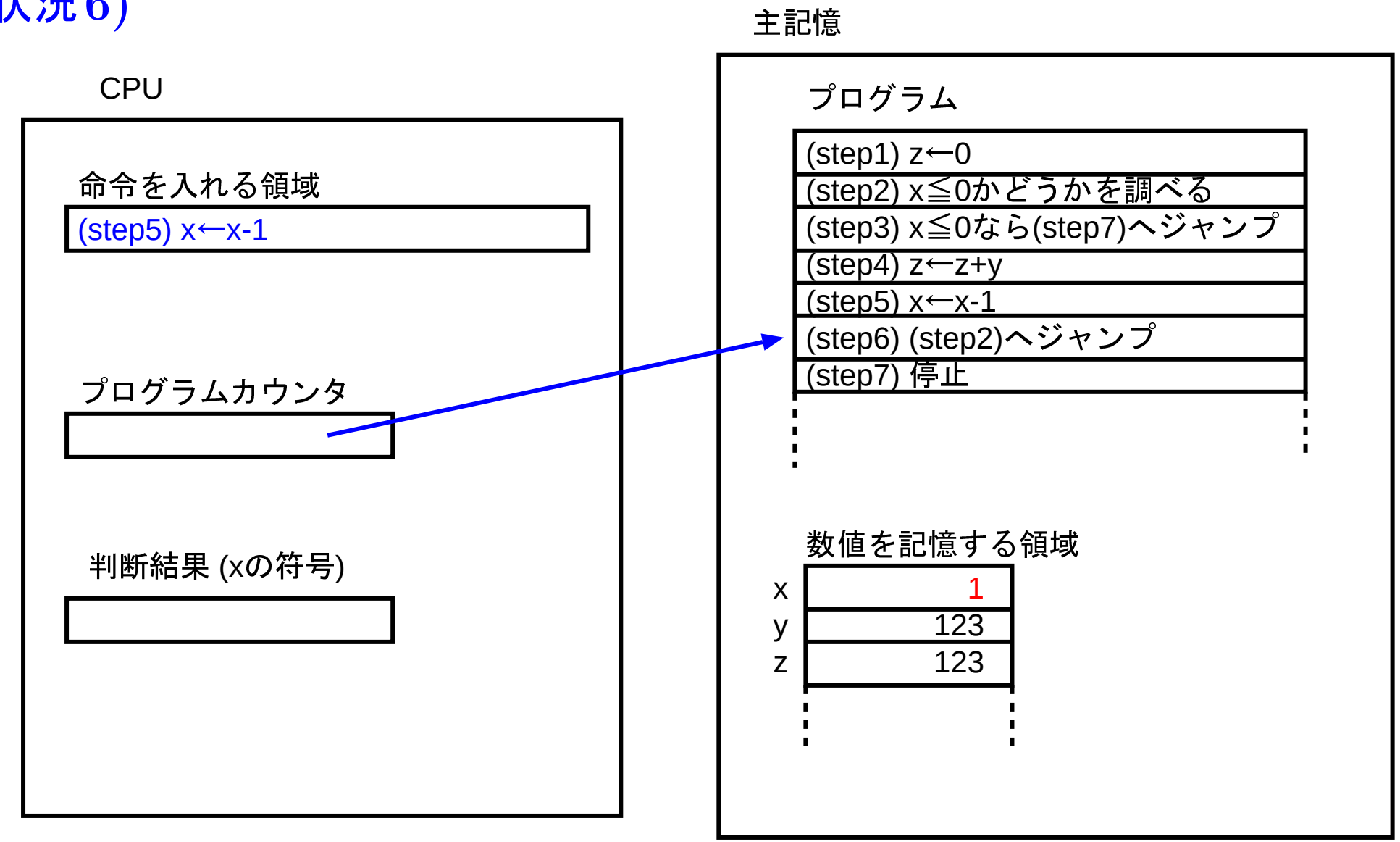

主記憶
CPU
プログラム
(step1) z←0
(step2) x≦0かどうかを調べる
(step3) x≦0なら(step7)へジャンプ
(step4) z←z+y
(step5) x←x-1
(step6) (step2)へジャンプ
(step7) 停止
命令を入れる領域
(step4) z←z+y
プログラムカウンタ
判断結果 (xの符号)
数値を記憶する領域
x 2
y 123
z 0

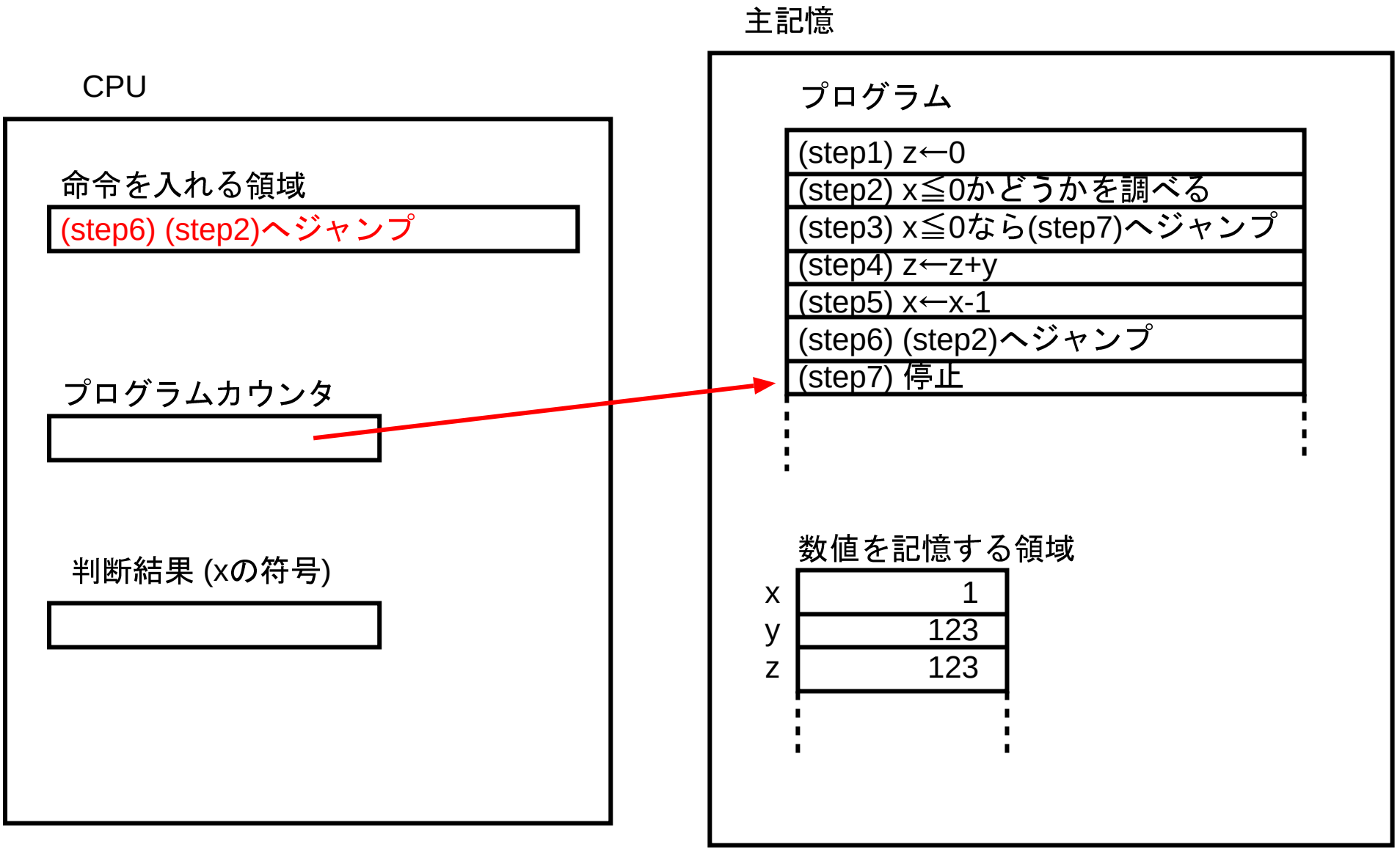
(状況5)


主記憶
CPU
命令を入れる領域
(step4) z←z+y
プログラムカウンタ
判断結果 (xの符号)
プログラム
(step1) z←0
(step2) x≦0かどうかを調べる
(step3) x≦0なら(step7)へジャンプ
(step4) z←z+y
(step5) x←x-1
(step6) (step2)へジャンプ
(step7) 停止
数値を記憶する領域
x 2
y 123
z 123

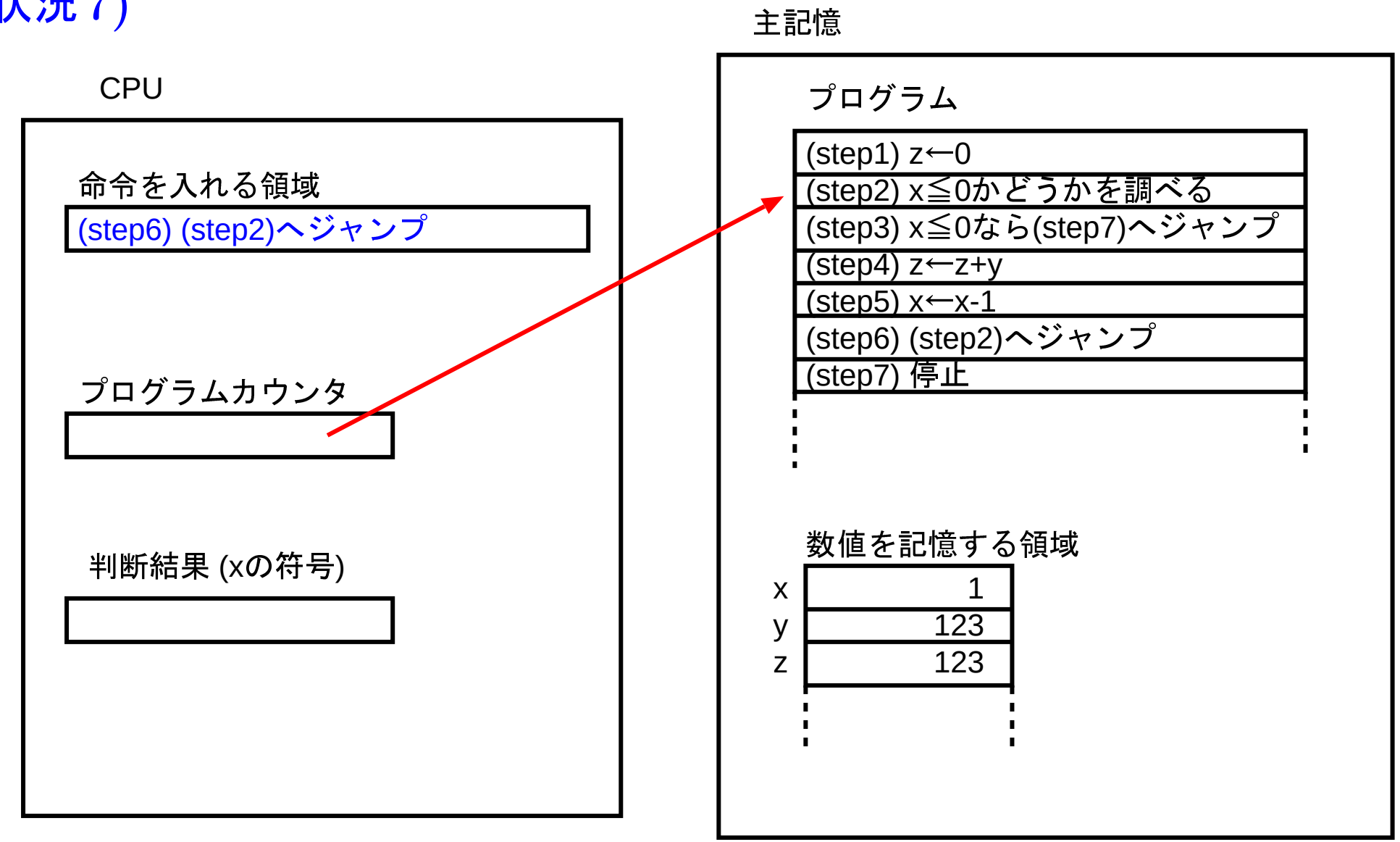

主記憶
CPU
プログラム
(step1) z←0
(step2) x≦0かどうかを調べる
(step3) x≦0なら(step7)へジャンプ
(step4) z←z+y
(step5) x←x-1
(step6) (step2)へジャンプ
(step7) 停止
命令を入れる領域
(step5) x←x-1
プログラムカウンタ
判断結果 (xの符号)
数値を記憶する領域
x 2
y 123
z 123

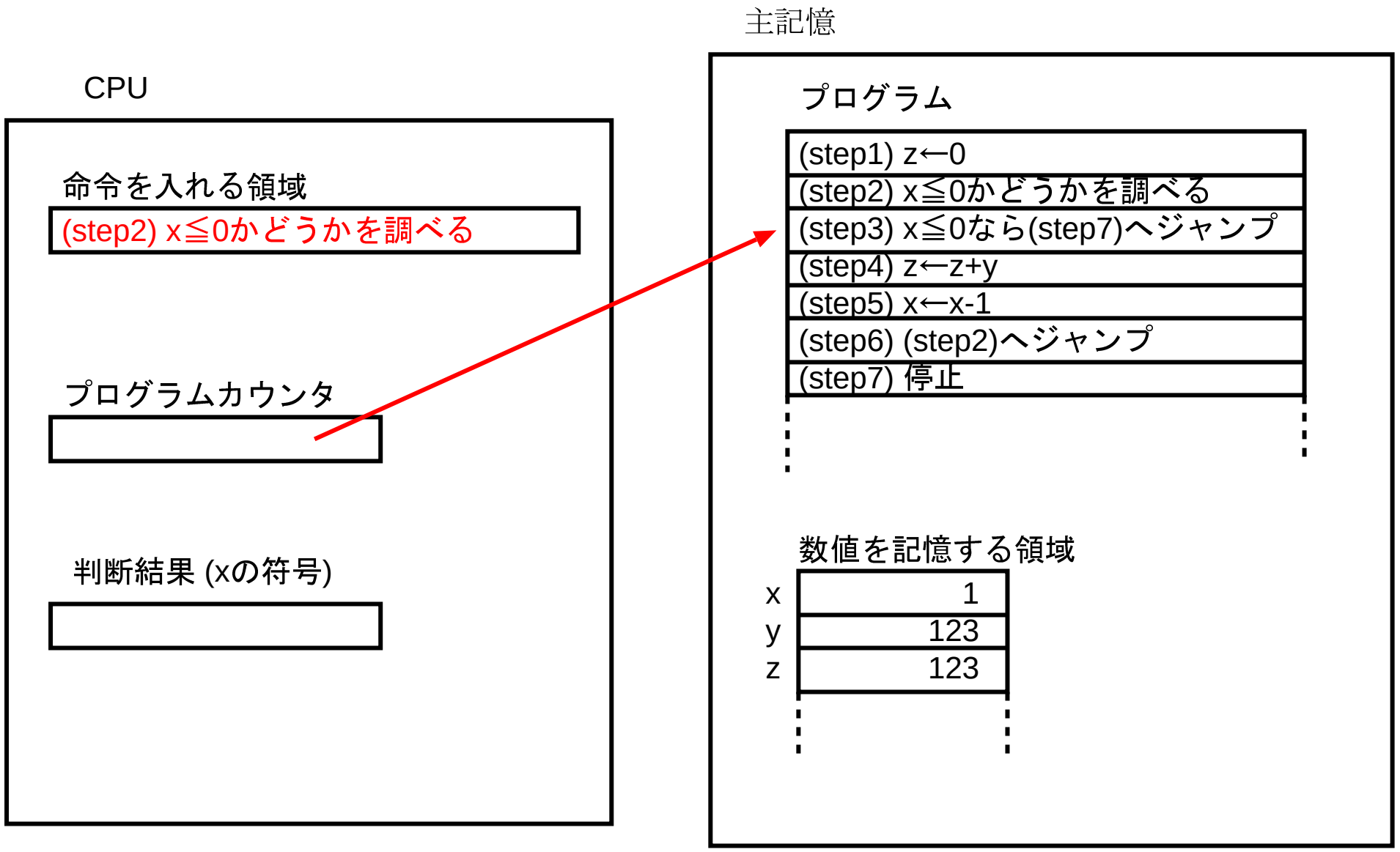
(状況6)


主記憶
CPU
プログラム
(step1) z←0
(step2) x≦0かどうかを調べる
(step3) x≦0なら(step7)へジャンプ
(step4) z←z+y
(step5) x←x-1
(step6) (step2)へジャンプ
(step7) 停止
命令を入れる領域
(step5) x←x-1
プログラムカウンタ
判断結果 (xの符号)
数値を記憶する領域
x
y
z
1
123
123

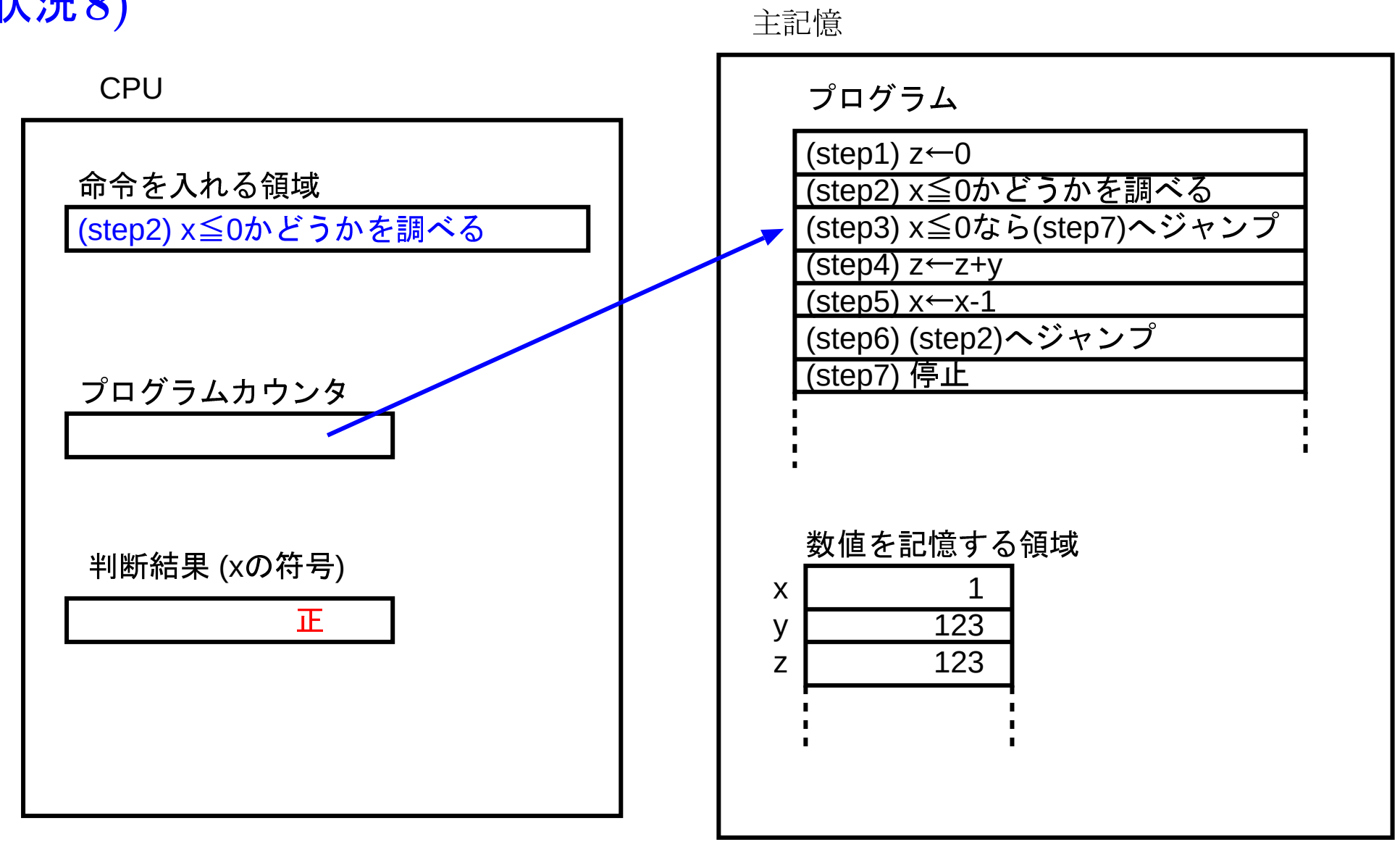

主記憶
CPU
プログラム
(step1) z←0
(step2) x≦0かどうかを調べる
(step3) x≦0なら(step7)へジャンプ
(step4) z←z+y
(step5) x←x-1
(step6) (step2)へジャンプ
(step7) 停止
命令を入れる領域
(step6) (step2)へジャンプ
プログラムカウンタ
判断結果 (xの符号)
数値を記憶する領域
x 1
y 123
z 123

(状況7)

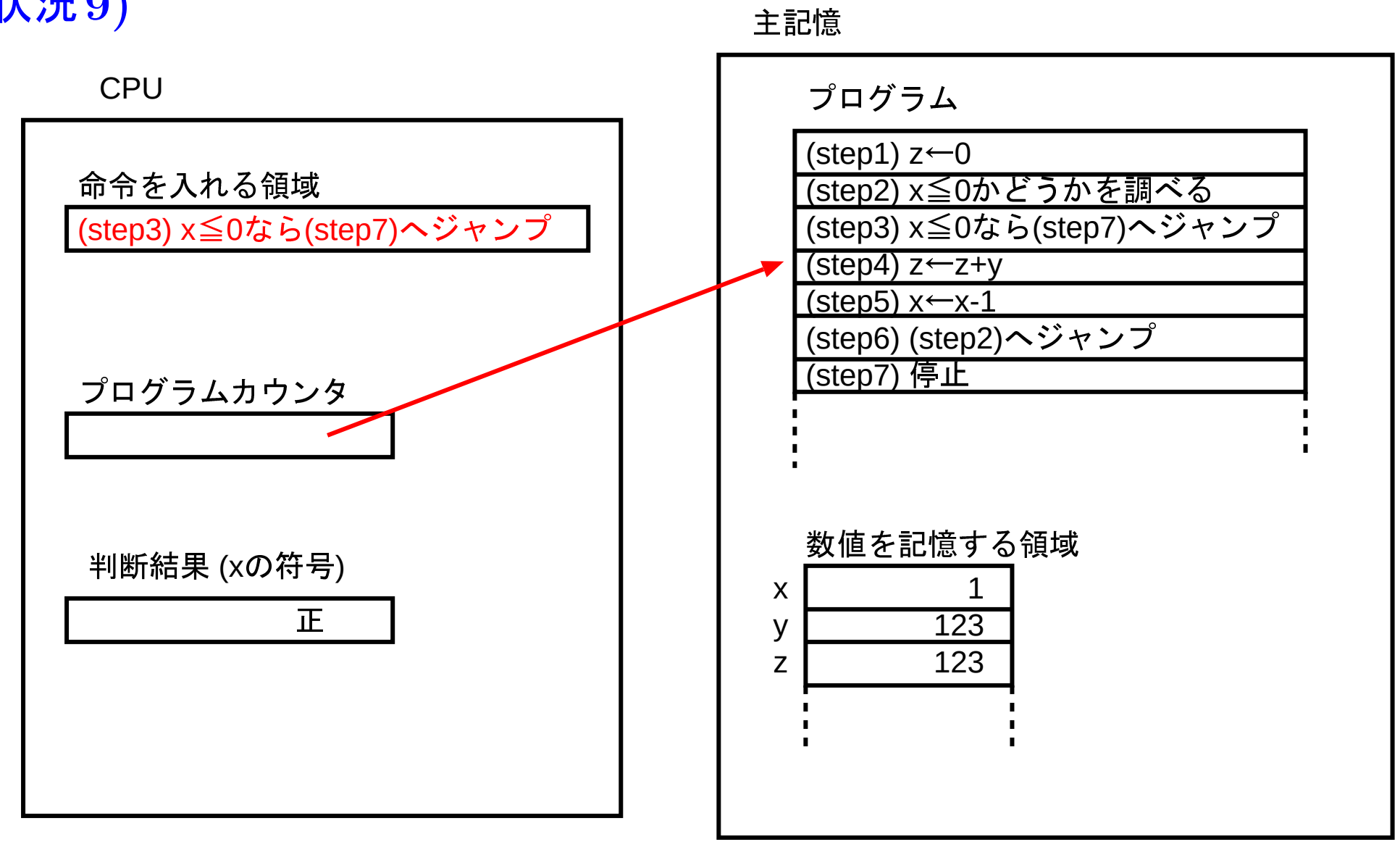

主記憶
CPU
命令を入れる領域
(step6) (step2)へジャンプ
プログラムカウンタ
判断結果 (xの符号)
プログラム
(step1) z←0
(step2) x≦0かどうかを調べる
(step3) x≦0なら(step7)へジャンプ
(step4) z←z+y
(step5) x←x-1
(step6) (step2)へジャンプ
(step7) 停止
数値を記憶する領域
x
y
z
1
123
123

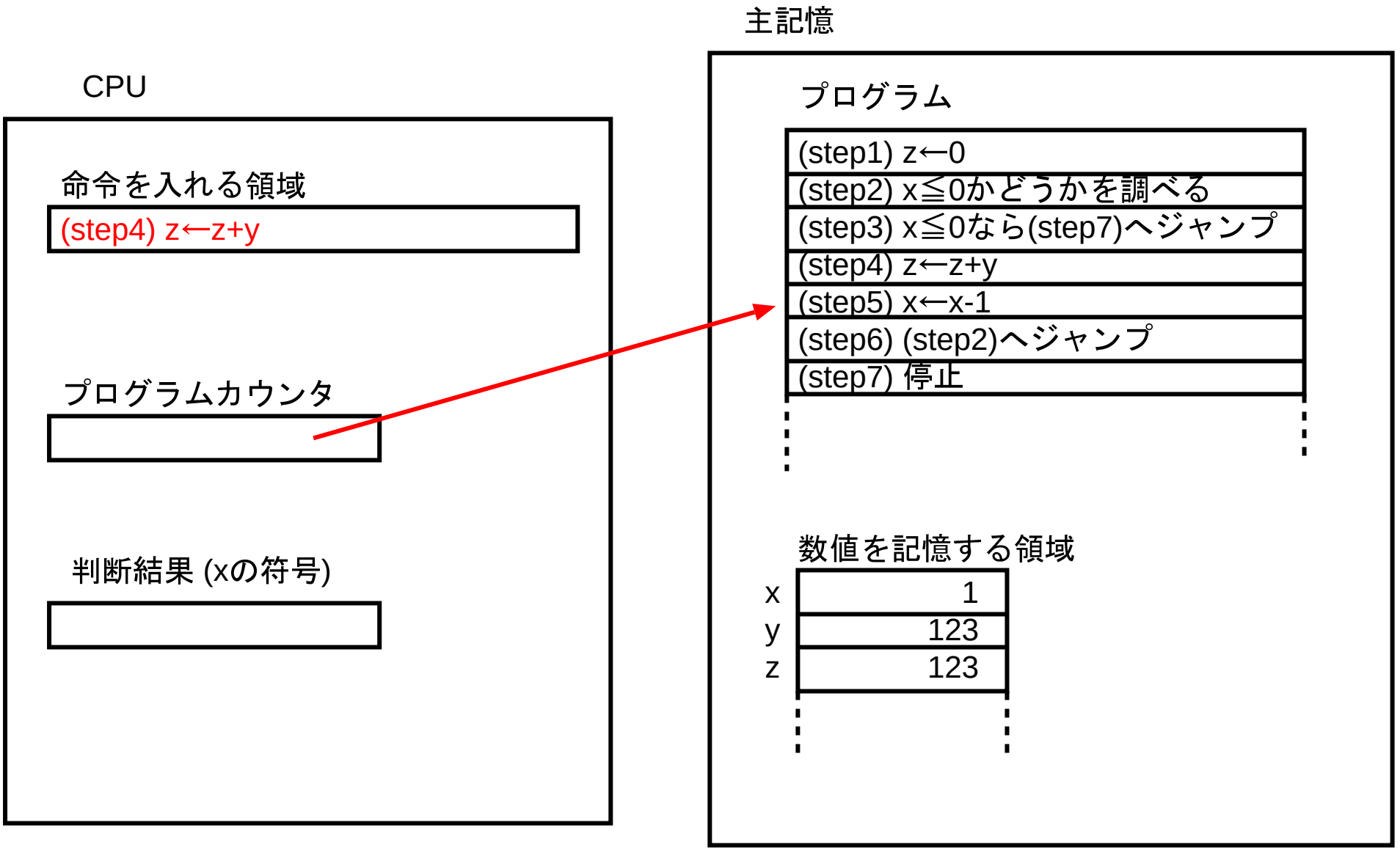

主記憶
CPU
プログラム
(step1) z←0
(step2) x≦0かどうかを調べる
(step3) x≦0なら(step7)へジャンプ
(step4) z←z+y
(step5) x←x-1
(step6) (step2)へジャンプ
(step7) 停止
命令を入れる領域
(step2) x≦0かどうかを調べる
プログラムカウンタ
判断結果 (xの符号)
数値を記憶する領域
x 1
y 123
z 123

(状況8)

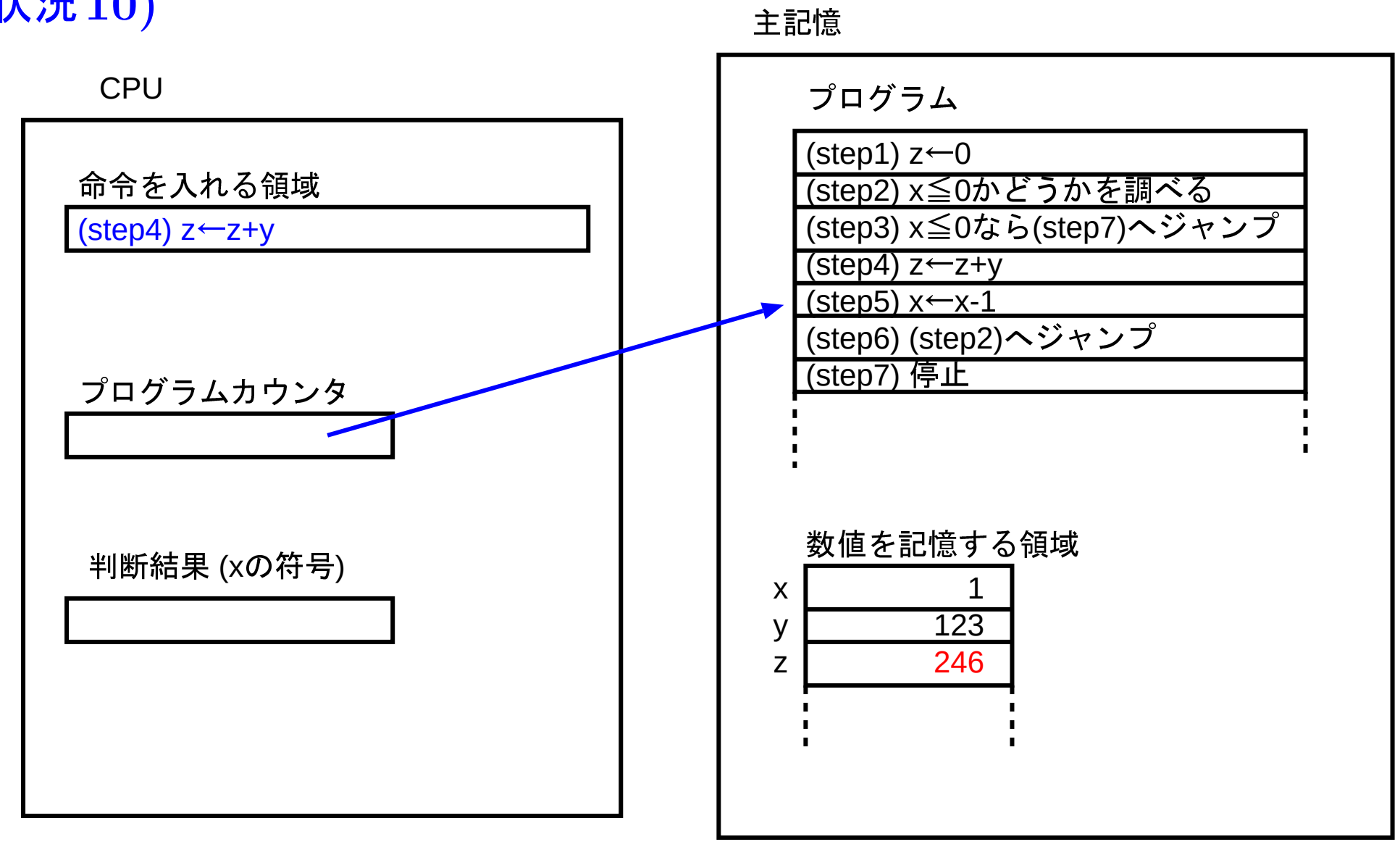

主記憶
CPU
命令を入れる領域
(step2) x≦0かどうかを調べる
プログラムカウンタ
判断結果 (xの符号)
正
プログラム
(step1) z←0
(step2) x≦0かどうかを調べる
(step3) x≦0なら(step7)へジャンプ
(step4) z←z+y
(step5) x←x-1
(step6) (step2)へジャンプ
(step7) 停止
数値を記憶する領域
x 1
y 123
z 123

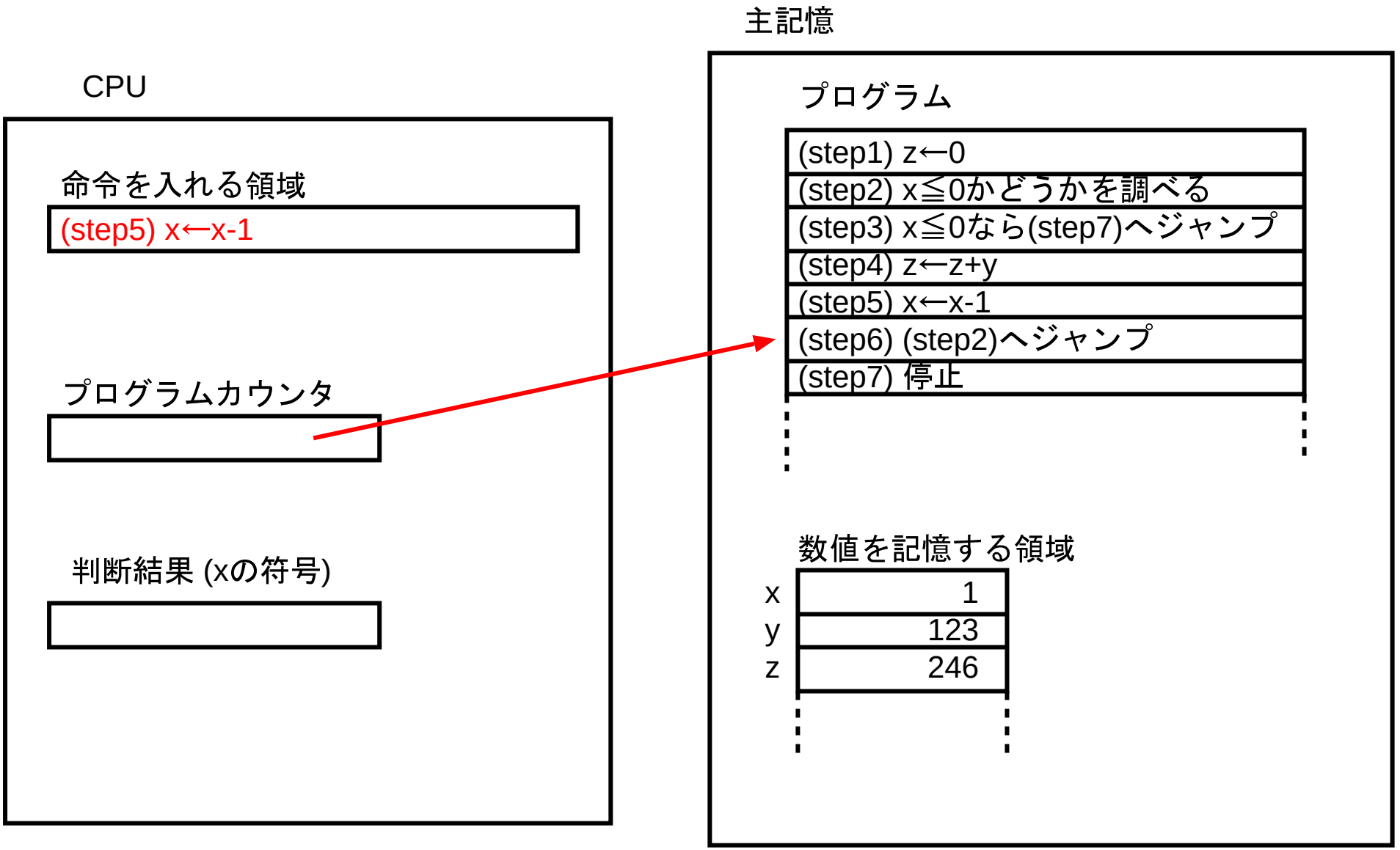
(状況9)


主記憶
CPU
命令を入れる領域
(step3) x≦0なら(step7)へジャンプ
プログラムカウンタ
判断結果 (xの符号)
正
プログラム
(step1) z←0
(step2) x≦0かどうかを調べる
(step3) x≦0なら(step7)へジャンプ
(step4) z←z+y
(step5) x←x-1
(step6) (step2)へジャンプ
(step7) 停止
数値を記憶する領域
x 1
y 123
z 123

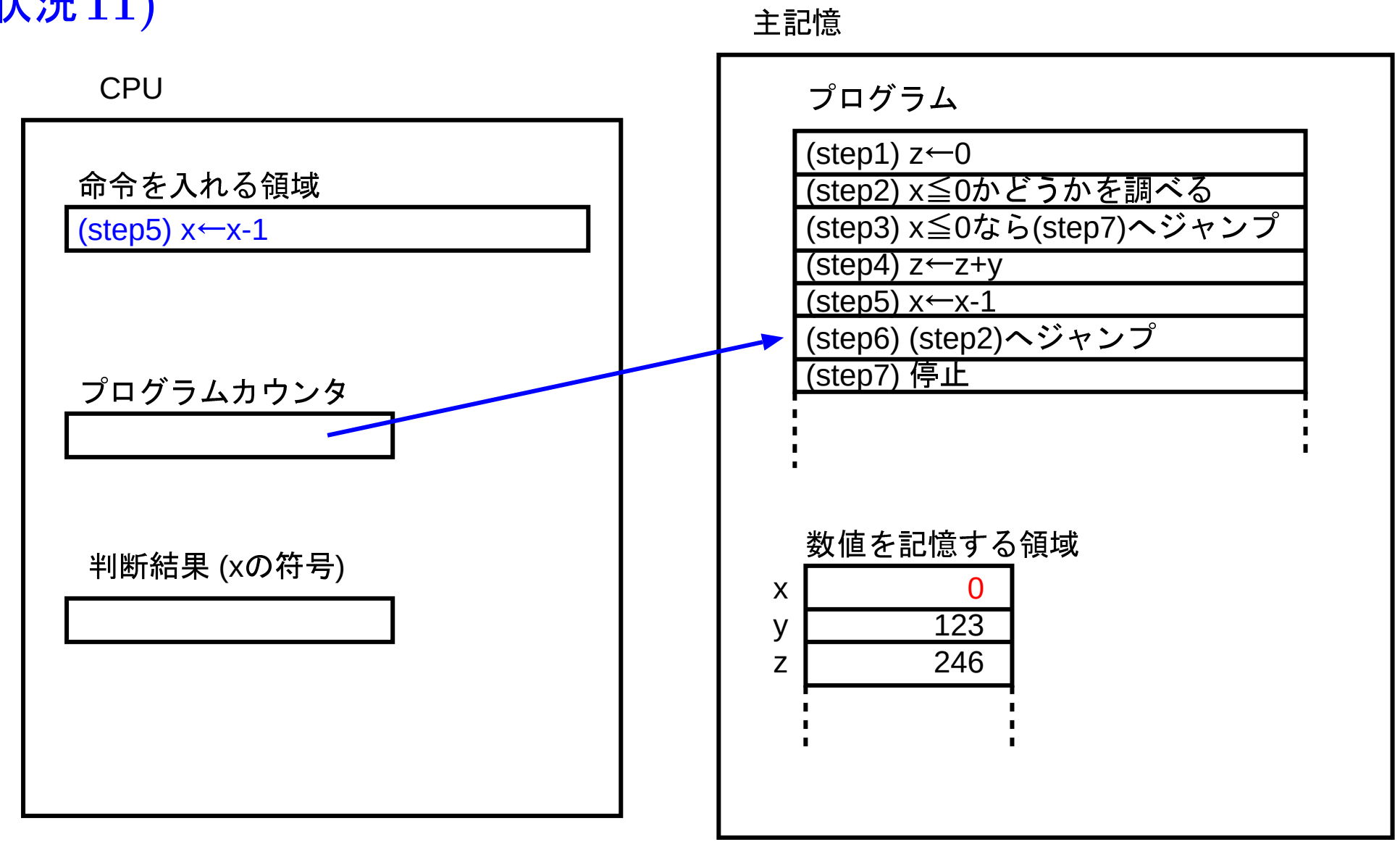

主記憶
CPU
プログラム
(step1) z←0
(step2) x≦0かどうかを調べる
(step3) x≦0なら(step7)へジャンプ
(step4) z←z+y
(step5) x←x-1
(step6) (step2)へジャンプ
(step7) 停止
命令を入れる領域
(step4) z←z+y
プログラムカウンタ
判断結果 (xの符号)
数値を記憶する領域
x 1
y 123
z 123

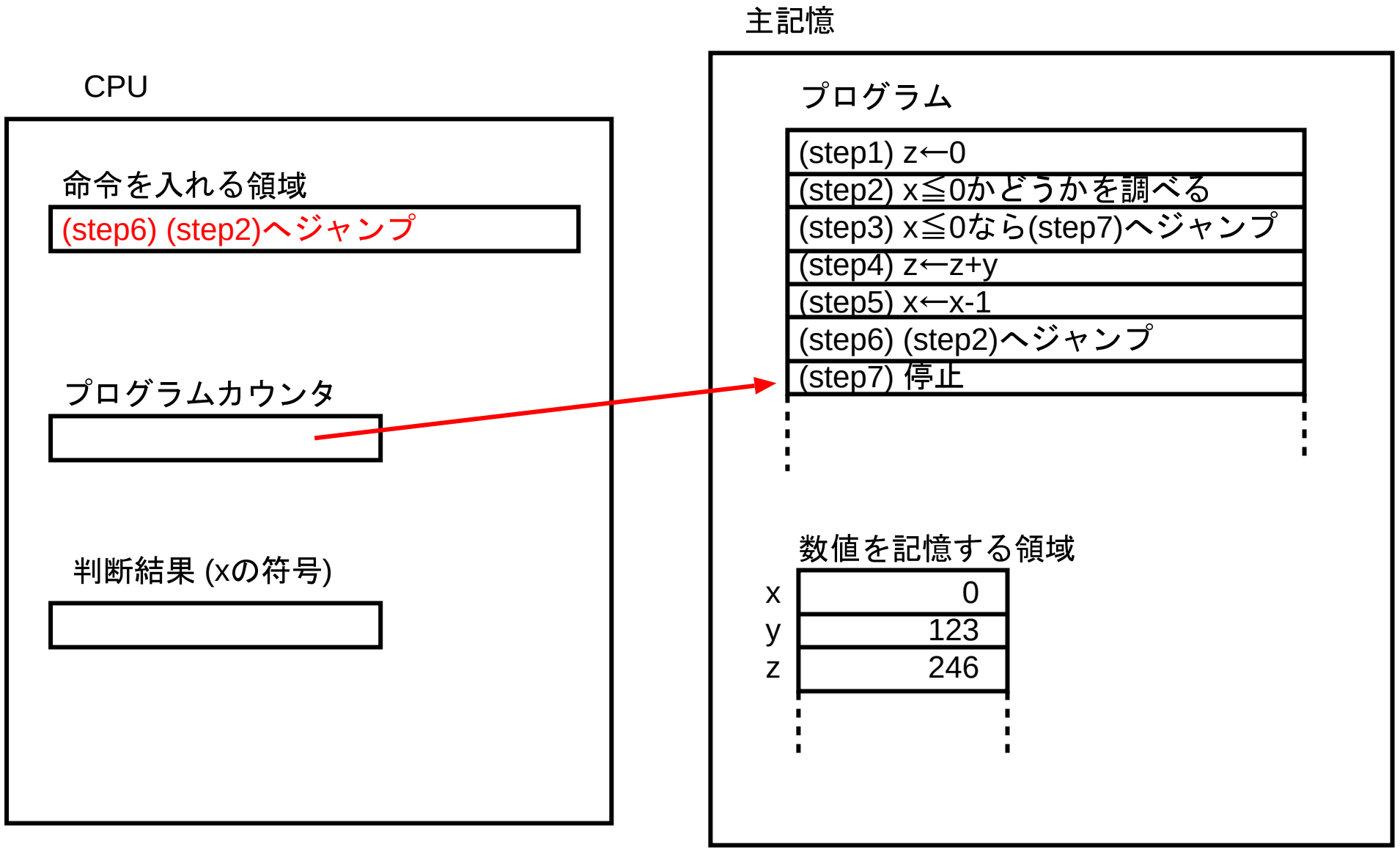
(状況10)


主記憶
CPU
命令を入れる領域
(step4) z←z+y
プログラムカウンタ
判断結果 (xの符号)
プログラム
(step1) z←0
(step2) x≦0かどうかを調べる
(step3) x≦0なら(step7)へジャンプ
(step4) z←z+y
(step5) x←x-1
(step6) (step2)へジャンプ
(step7) 停止
数値を記憶する領域
x
y
z
1
123
246

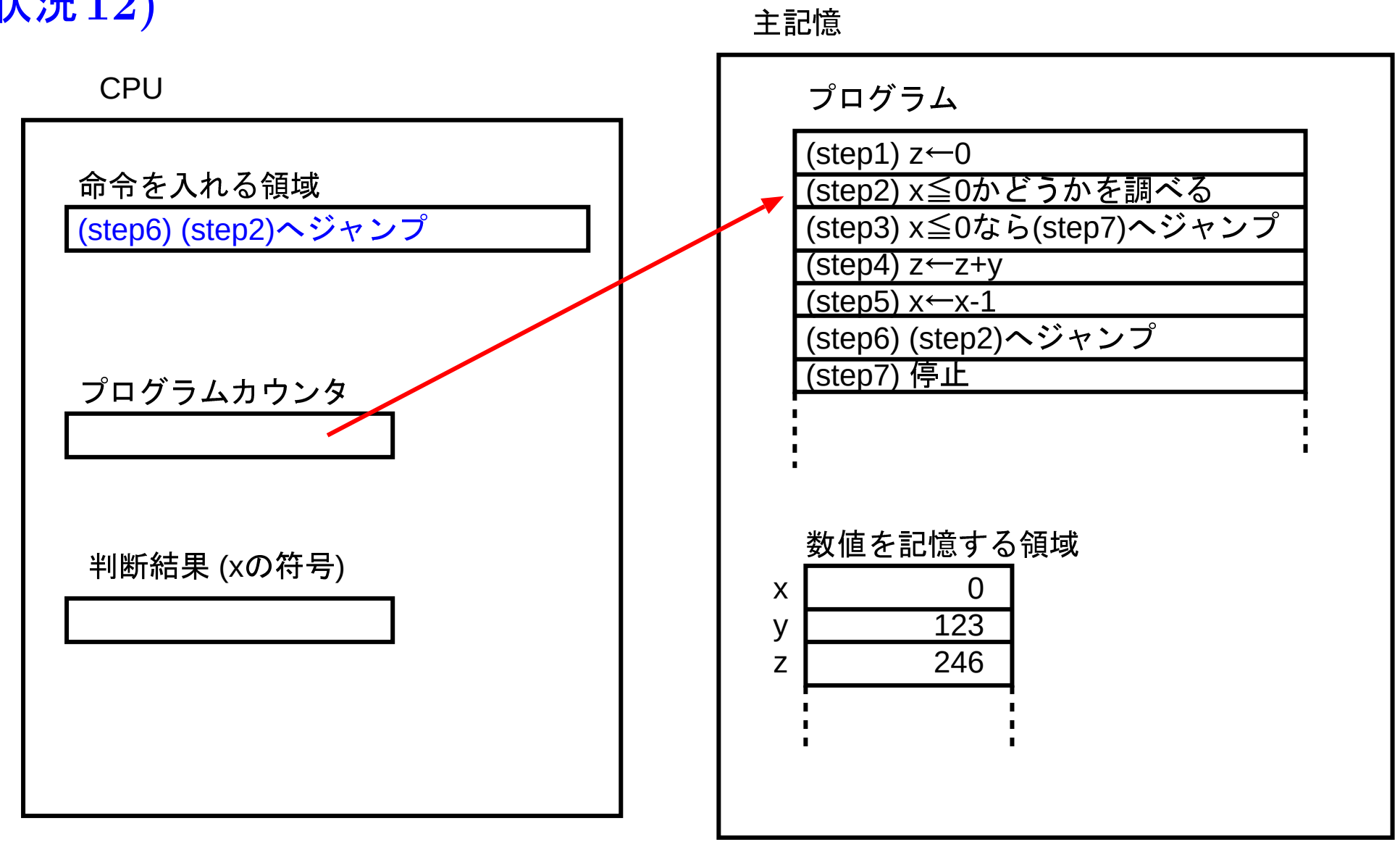

主記憶
CPU
プログラム
(step1) z←0
(step2) x≦0かどうかを調べる
(step3) x≦0なら(step7)へジャンプ
(step4) z←z+y
(step5) x←x-1
(step6) (step2)へジャンプ
(step7) 停止
命令を入れる領域
(step5) x←x-1
プログラムカウンタ
判断結果 (xの符号)
数値を記憶する領域
x 1
y 123
z 246

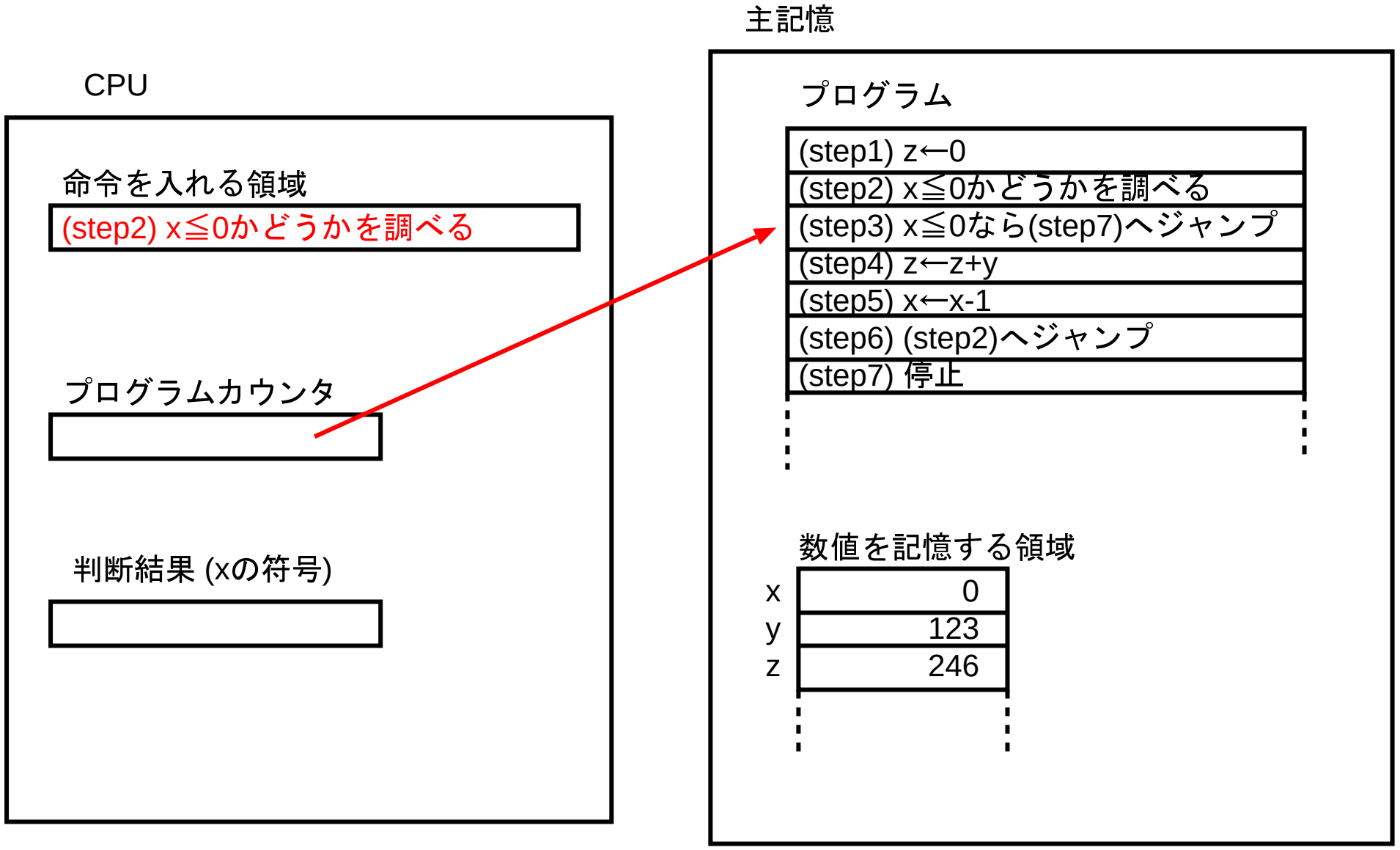
(状況11)


主記憶
CPU
プログラム
(step1) z←0
(step2) x≦0かどうかを調べる
(step3) x≦0なら(step7)へジャンプ
(step4) z←z+y
(step5) x←x-1
(step6) (step2)へジャンプ
(step7) 停止
命令を入れる領域
(step5) x←x-1
プログラムカウンタ
判断結果 (xの符号)
数値を記憶する領域
x
y
z
0
123
246

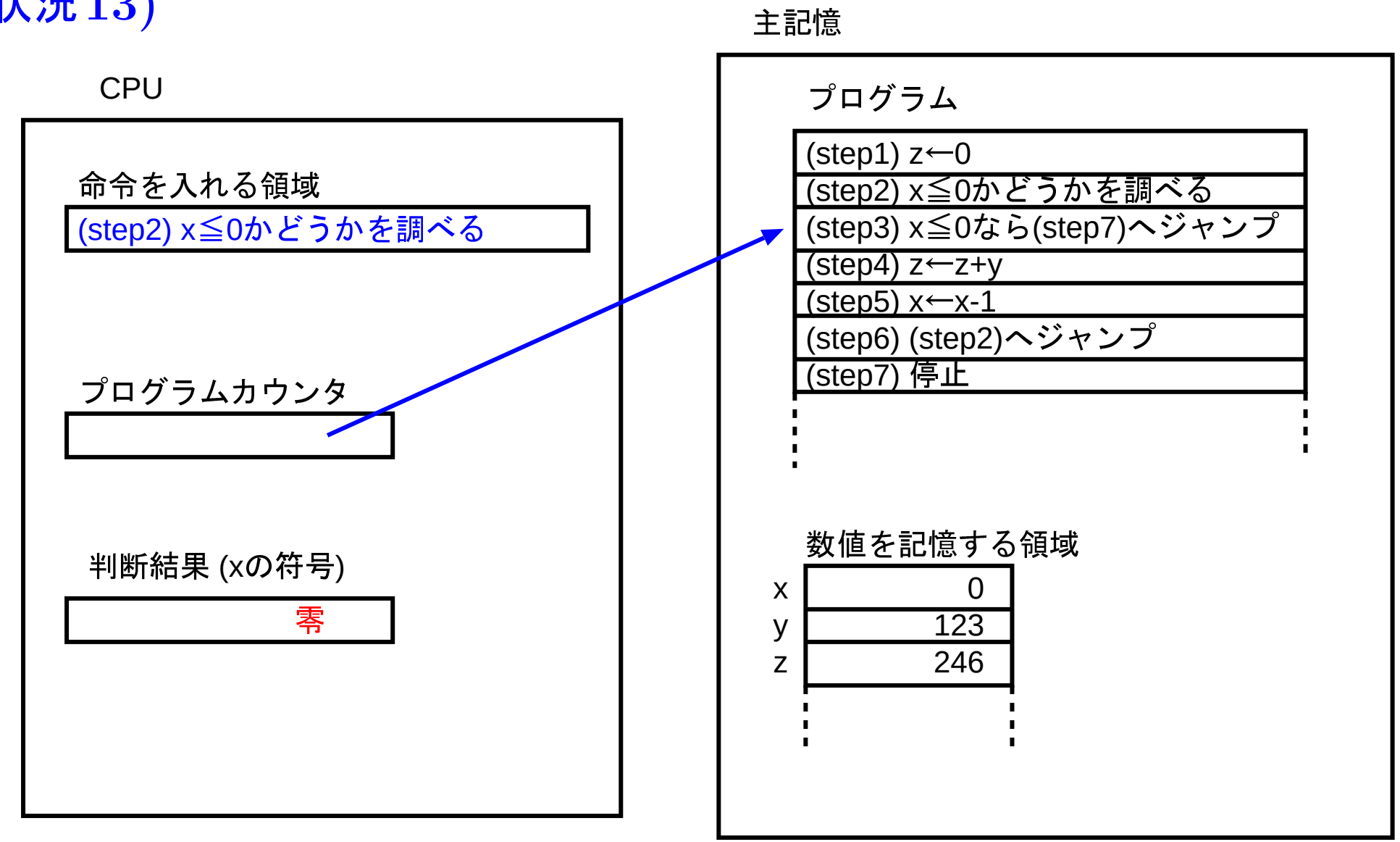

主記憶
CPU
プログラム
(step1) z←0
(step2) x≦0かどうかを調べる
(step3) x≦0なら(step7)へジャンプ
(step4) z←z+y
(step5) x←x-1
(step6) (step2)へジャンプ
(step7) 停止
命令を入れる領域
(step6) (step2)へジャンプ
プログラムカウンタ
判断結果 (xの符号)
数値を記憶する領域
x
y
z
0
123
246

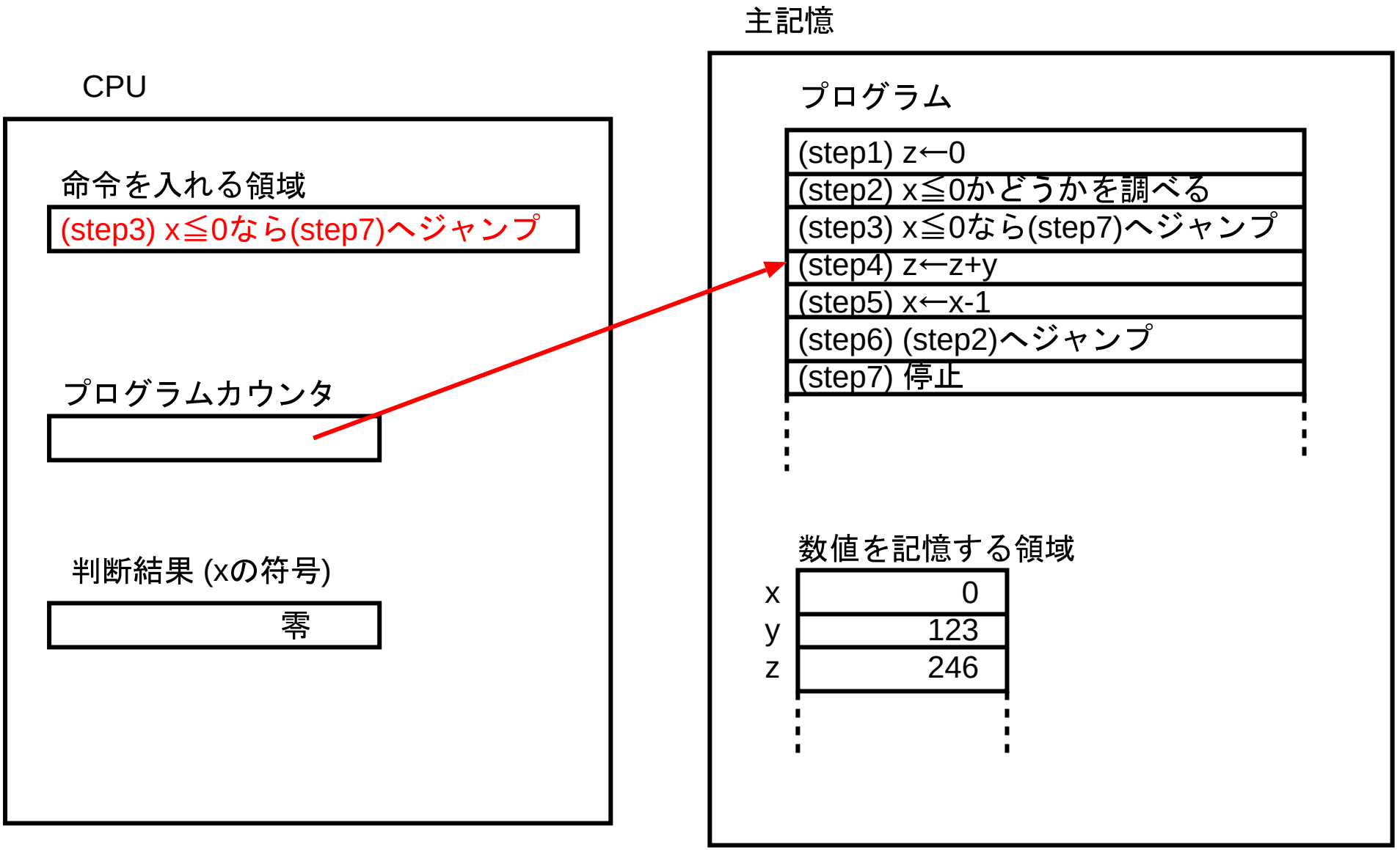
(状況12)


主記憶
CPU
プログラム
(step1) z←0
(step2) x≦0かどうかを調べる
(step3) x≦0なら(step7)へジャンプ
(step4) z←z+y
(step5) x←x-1
(step6) (step2)へジャンプ
(step7) 停止
命令を入れる領域
(step6) (step2)へジャンプ
プログラムカウンタ
判断結果 (xの符号)
数値を記憶する領域
x
y
z
0
123
246

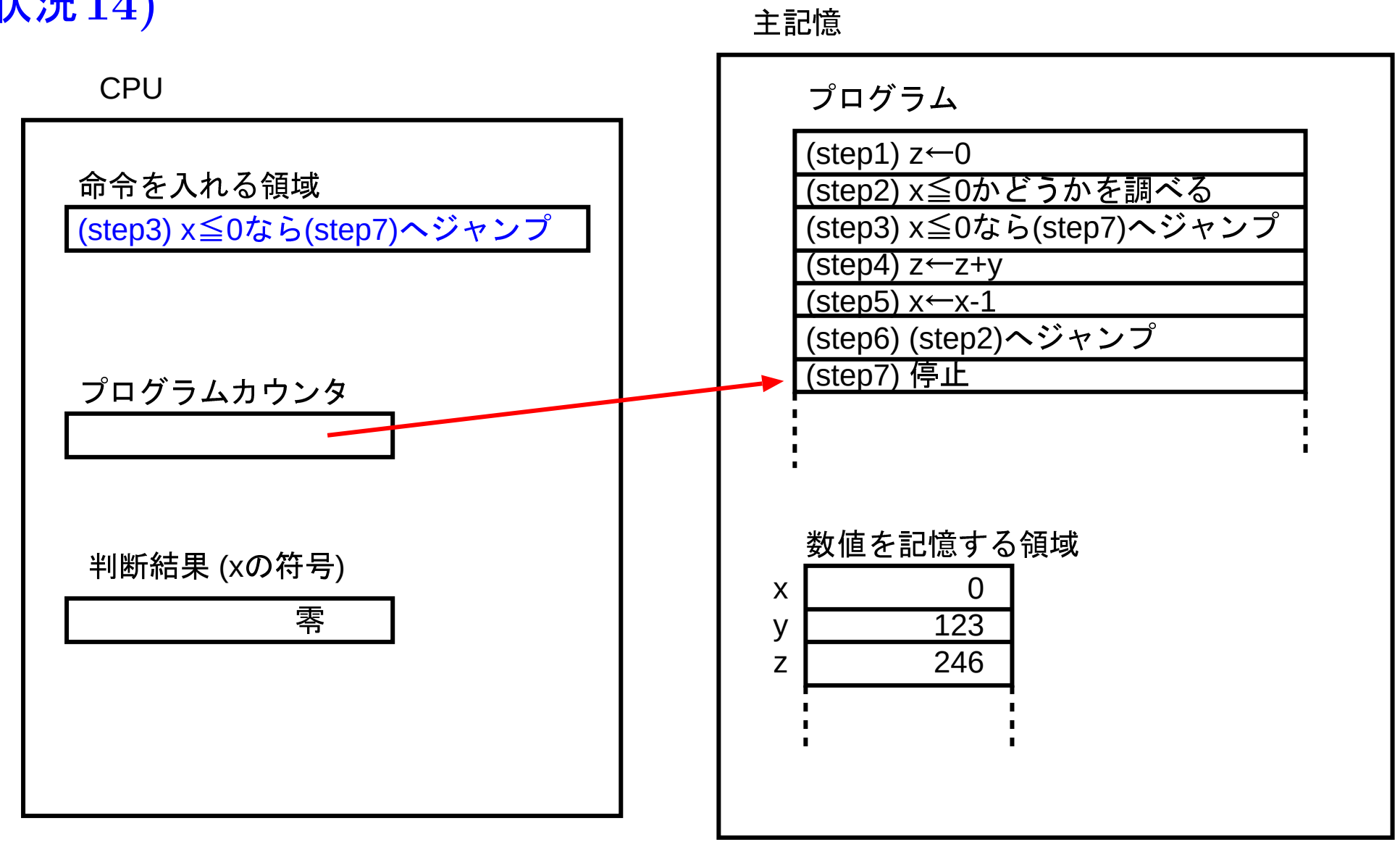

主記憶
CPU
プログラム
(step1) z←0
(step2) x≦0かどうかを調べる
(step3) x≦0なら(step7)へジャンプ
(step4) z←z+y
(step5) x←x-1
(step6) (step2)へジャンプ
(step7) 停止
命令を入れる領域
(step2) x≦0かどうかを調べる
プログラムカウンタ
判断結果 (xの符号)
数値を記憶する領域
x 0
y 123
z 246

(状況13)


主記憶
CPU
プログラム
(step1) z←0
(step2) x≦0かどうかを調べる
(step3) x≦0なら(step7)へジャンプ
(step4) z←z+y
(step5) x←x-1
(step6) (step2)へジャンプ
(step7) 停止
命令を入れる領域
(step2) x≦0かどうかを調べる
プログラムカウンタ
判断結果 (xの符号)
零
数値を記憶する領域
x
y
z
0
123
246


主記憶
CPU
プログラム
(step1) z←0
(step2) x≦0かどうかを調べる
(step3) x≦0なら(step7)へジャンプ
(step4) z←z+y
(step5) x←x-1
(step6) (step2)へジャンプ
(step7) 停止
命令を入れる領域
(step3) x≦0なら(step7)へジャンプ
プログラムカウンタ
判断結果 (xの符号)
零
数値を記憶する領域
x
y
z
0
123
246

(状況14)


主記憶
CPU
プログラム
(step1) z←0
(step2) x≦0かどうかを調べる
(step3) x≦0なら(step7)へジャンプ
(step4) z←z+y
(step5) x←x-1
(step6) (step2)へジャンプ
(step7) 停止
命令を入れる領域
(step3) x≦0なら(step7)へジャンプ
プログラムカウンタ
判断結果 (xの符号)
零
数値を記憶する領域
x 0
y 123
z 246