計算機科学実験及演習３ ハードウェア
SIMPLEアーキテクチャのプロセッサの実装

京都大学 工学部情報学科 計算機科学コース

計算機科学実験及演習３

ハードウェア担当

Ver.2025/04

この講義の内容

- この講義は本実験の課題とSIMPLEの仕様の説明を目的としています。
- マイクロプロセッサ自体の構成や動作原理などは理解しているものとして説明をします。
- 実験HPのSIMPLE設計資料に沿って説明しますので、そちらも参考にしてください。

実験３ハードウェアの内容と目的

- 内容
  - SIMPLEをベースとした独自のマイクロプロセッサの設計（方式設計、論理設計）とFPGAでの実装

目的

  - プロセッサの動作原理を理解する
  - 回路設計、最適化、動作テストの方法を習得する
  - プロセッサの種々の拡張方式や最適化技術を実践的に学ぶ

- 参考文献
  - 富田眞治、中島浩： コンピュータハードウェア
  - D.A.パターソン、J.L.ヘネシー著、成田光彰訳：
    コンピュータの構成と設計(上),(下)　　など，，，

SIMPLEアーキテクチャの仕様

SIMPLEアーキテクチャの概要

SIxteen-bit MicroProcessor for Laboratory Experiment

- 😐簡単な命令セット
- ☺基本機能は1通り備えられている

- 特徴
  - 16bit固定長命令
  - 16bit×64K語の主記憶
  - ８本の汎用レジスタ
  - 2オペランド形式の命令セット(Rd op Rs -> Rd)

主記憶とレジスタ

1. 主記憶
   - 16bit×64K語 （語アドレス方式）
   - ただし、実験で使用するFPGAボードで確保できる最大サイズは約33K語
2. 汎用レジスタ
   - 16bit×8語
3. プログラムカウンタ (PC)
   - 16bit
4. 条件コード
   - S　サイン
   - Z　ゼロ
   - C　キャリー
   - V　オーバーフロー

命令セット

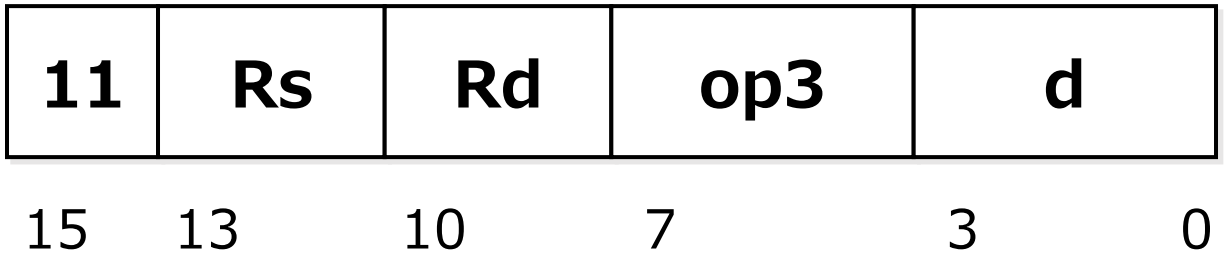
命令はすべて16bit固定長。以下の命令形式がある。

1. 演算／入出力命令
2. ロード／ストア命令
3. 即値ロード／無条件分岐命令
4. 条件分岐命令

1.演算命令

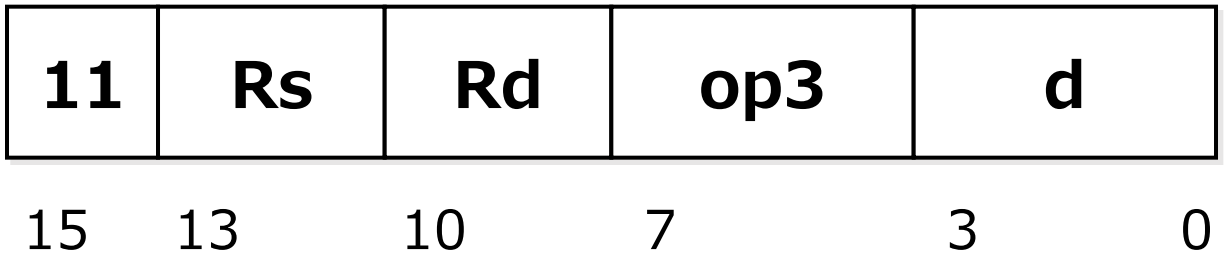
Rs, Rdで指定されたレジスタに格納されている値にop3の演算を施してRdに格納する。

- 算術論理演算命令
  - **r[Rd] = r[Rd] op3 r[Rs]**
- シフト命令
  - **r[Rd] = shift_op3(r[Rd], d)**
- 注：実行後に条件コードをセットする



1.入出力命令

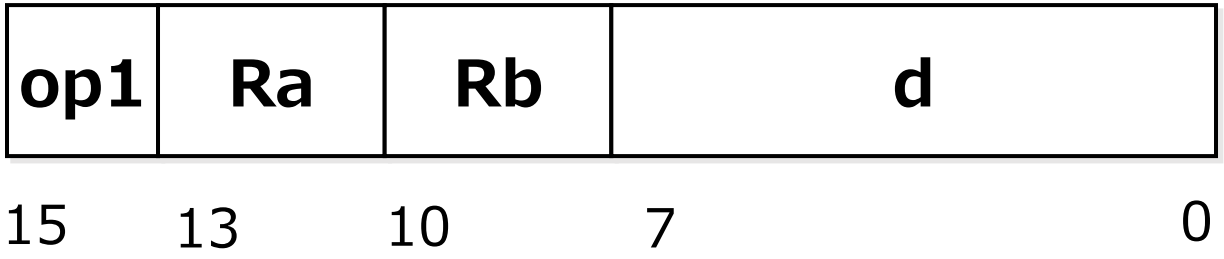
- 入力命令**(op3: 1100)**
  - **r[Rd] = input**
  - 入力先はボード上のスイッチ など
- 出力命令**(op3: 1101)**
  - **output = r[Rs]**
  - 出力先はボードのLED/7SEG LED など



2.ロード／ストア命令

Rbレジスタの値に、dフィールドの値を加算した値をアドレスとしてメモリにアクセスする（ベースレジスタアドレス指定）

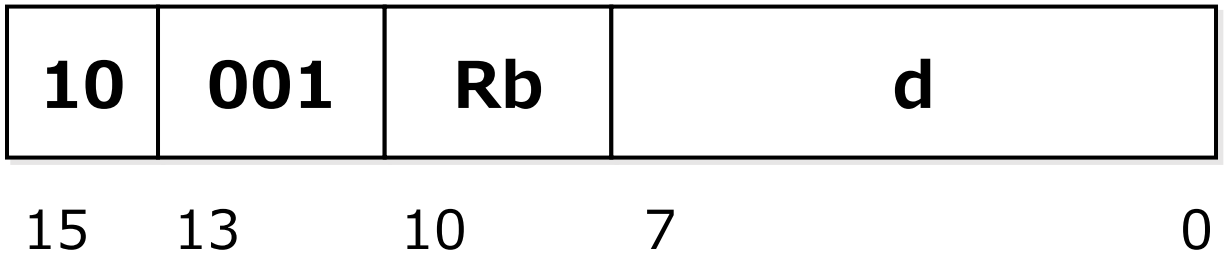
- ロード命令 **(op1 : 00)**
  - **r[Ra] = *(r[Rb] + sign_ext(d))**
- ストア命令 **(op1 : 01)**
  - ***(r[Rb] + sign_ext(d)) = r[Ra]**



3.即値ロード命令

dフィールドの値を16bitに拡張した値をRbレジスタに格納する。

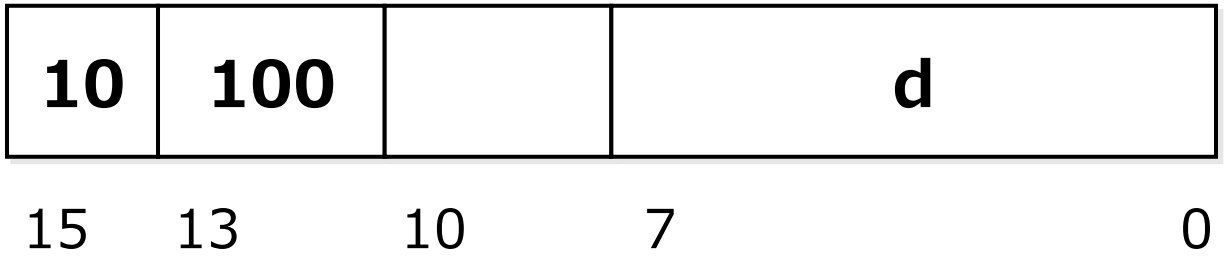
- 即値ロード命令
  - **r[Rb] = sign_ext(d)**



3.無条件分岐命令

PC(プログラムカウンタ)の値に符号拡張したdフィールドの値を加算する。次に実行される命令を分岐する。

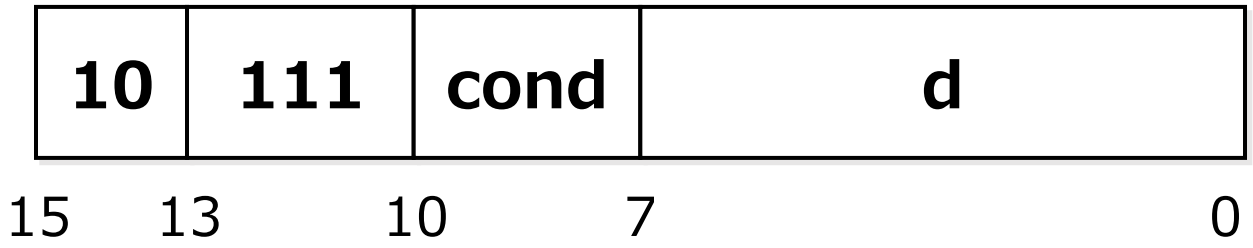
- 無条件分岐命令(B: Branch)
  - **PC = PC + 1 + sign_ext(d)**



4.条件分岐命令

condに定められている条件が成り立つ場合に、PCの値に符号拡張したdフィールドの値を加算し、分岐させる。

- 条件分岐命令
  - **if (cond) PC = PC + 1 + sign_ext(d)**
  - 条件コードの値に従って分岐
    - 条件コードは演算命令の実行時にセットされる

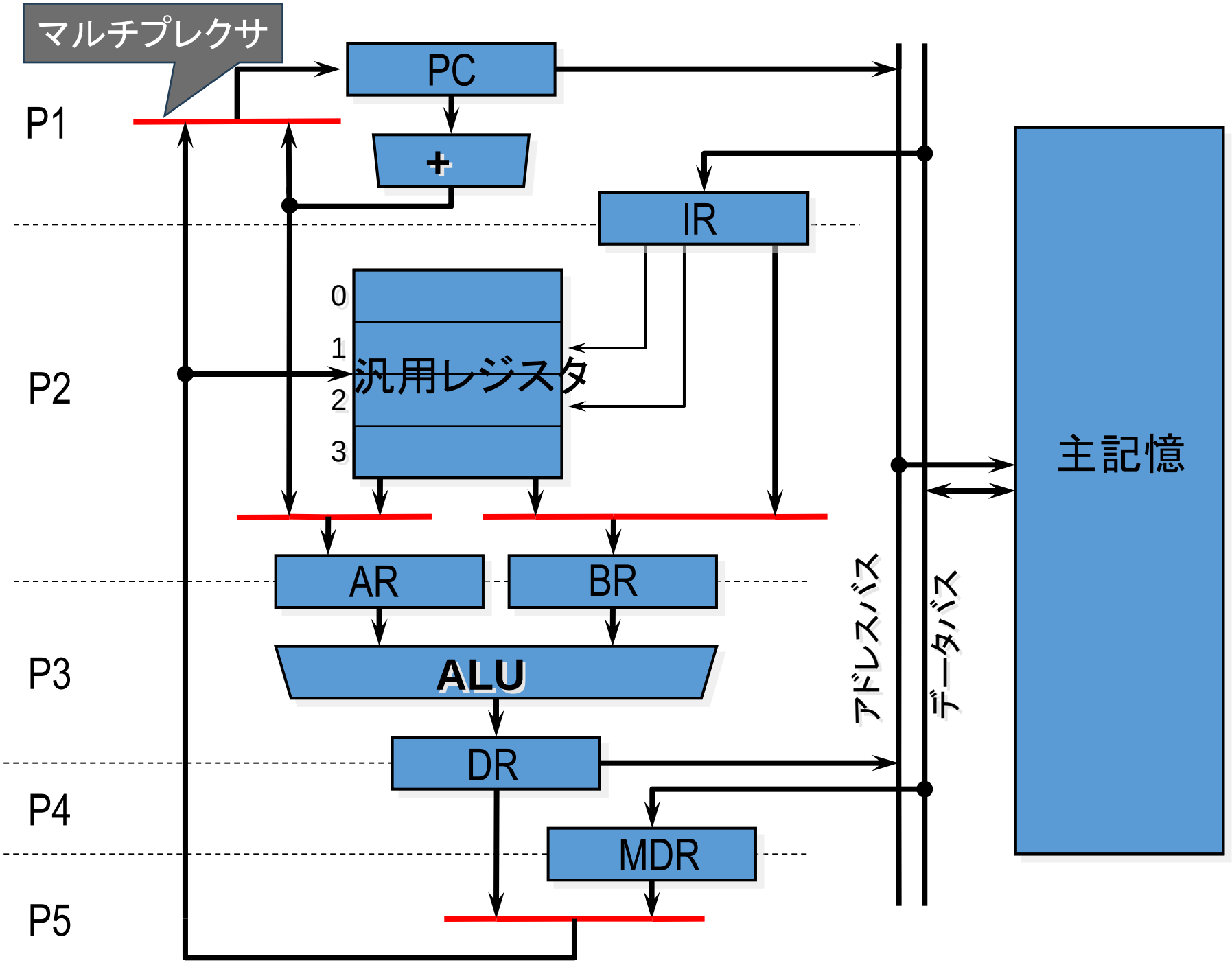


設計と動作の例

基本的な設計　SIMPLE/B

- 命令実行の過程を5つのフェーズに分割。逐次活性化
  - P1　　命令フェッチ
  - P2　　命令デコード、レジスタ読み出し
  - P3　　演算
  - P4　　主記憶アクセス
  - P5　　レジスタ書き込み／PC更新
- レジスタ（IR、AR・BR、DR、MDR）
  - 各フェーズでの値を保持
- 制御回路
  - プロセッサを構成するすべてのモジュールの制御
    - マルチプレクサを適切に切り替える
    - フェーズから出力されるデータを保持するレジスタを更新
    - ALUの機能の選択
    - などなど

次スライドにプロセッサを構成する要素を配置したブロック図を表示（紙面の都合上、簡略化）


マルチプレクサ
PC
P1
+
IR
0
1
2
3
汎用レジスタ
P2
主記憶
AR
BR
P3
ALU
アドレスバス
データバス
DR
P4
MDR
P5

命令実行の例

CPU上で以下の命令が実行されるときの、CPUの動作を示す。

```
LD R0, 10(R1)
ADD R0, R2
B -5
```

R1レジスタに格納されている値と10を足した値をアドレスとしてメモリから値を読み出しR0レジスタに格納。(ロード)

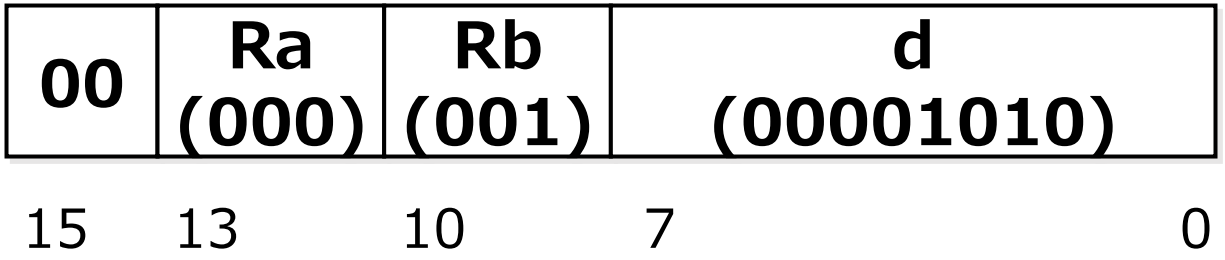
R2レジスタの値とR0レジスタの値を足して、R0レジスタに格納。(加算)

プログラムカウンタを現在のプログラムカウンタの値に-5を加算した値にする。(無条件分岐)

命令実行の例

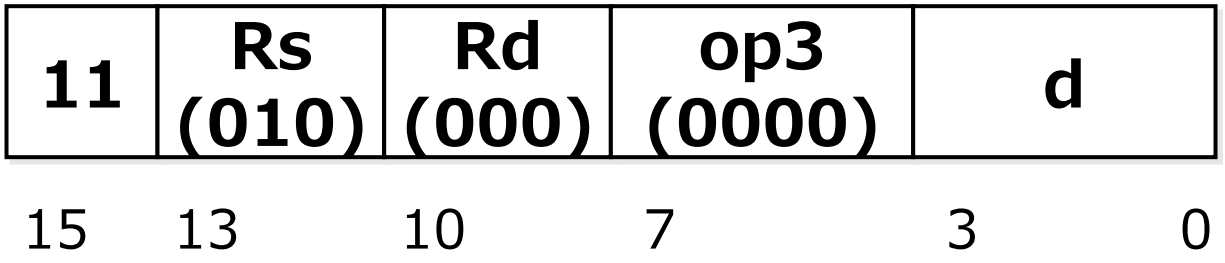
- ロード命令: プログラムカウンタ100
  - LD R0, 10(R1)

略記 0 0 1 10



- 加算命令： プログラムカウンタ101
  - ADD R0, R2

略記 3 2 0 0 -

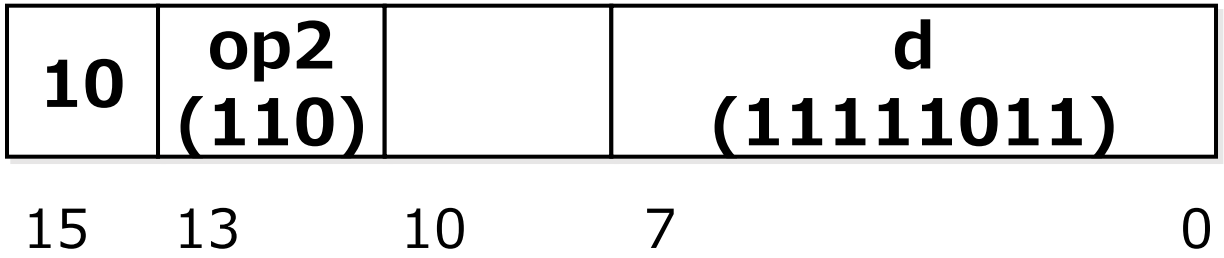


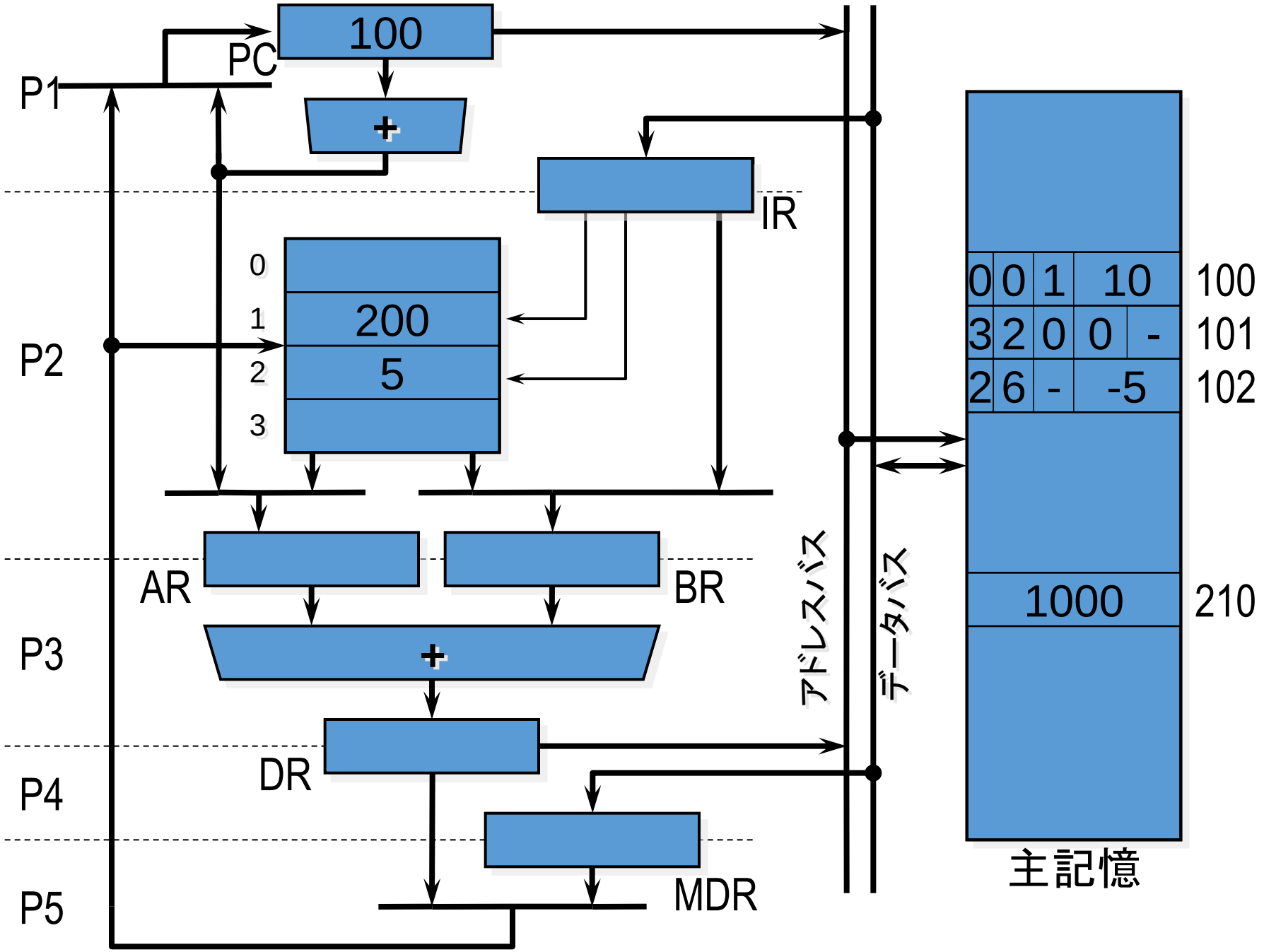
命令実行の例

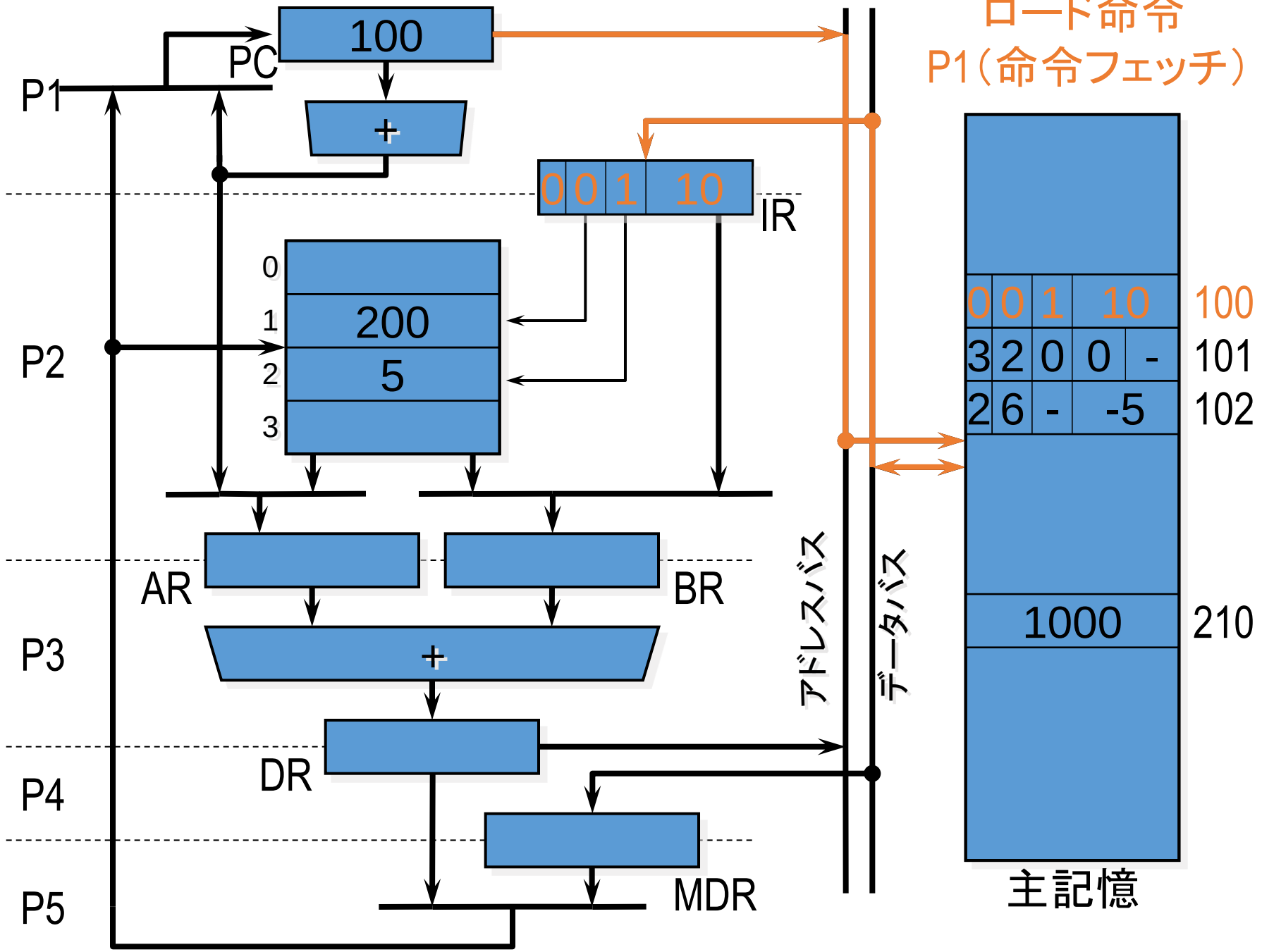
- 無条件分岐命令：プログラムカウンタ102
  - B -5

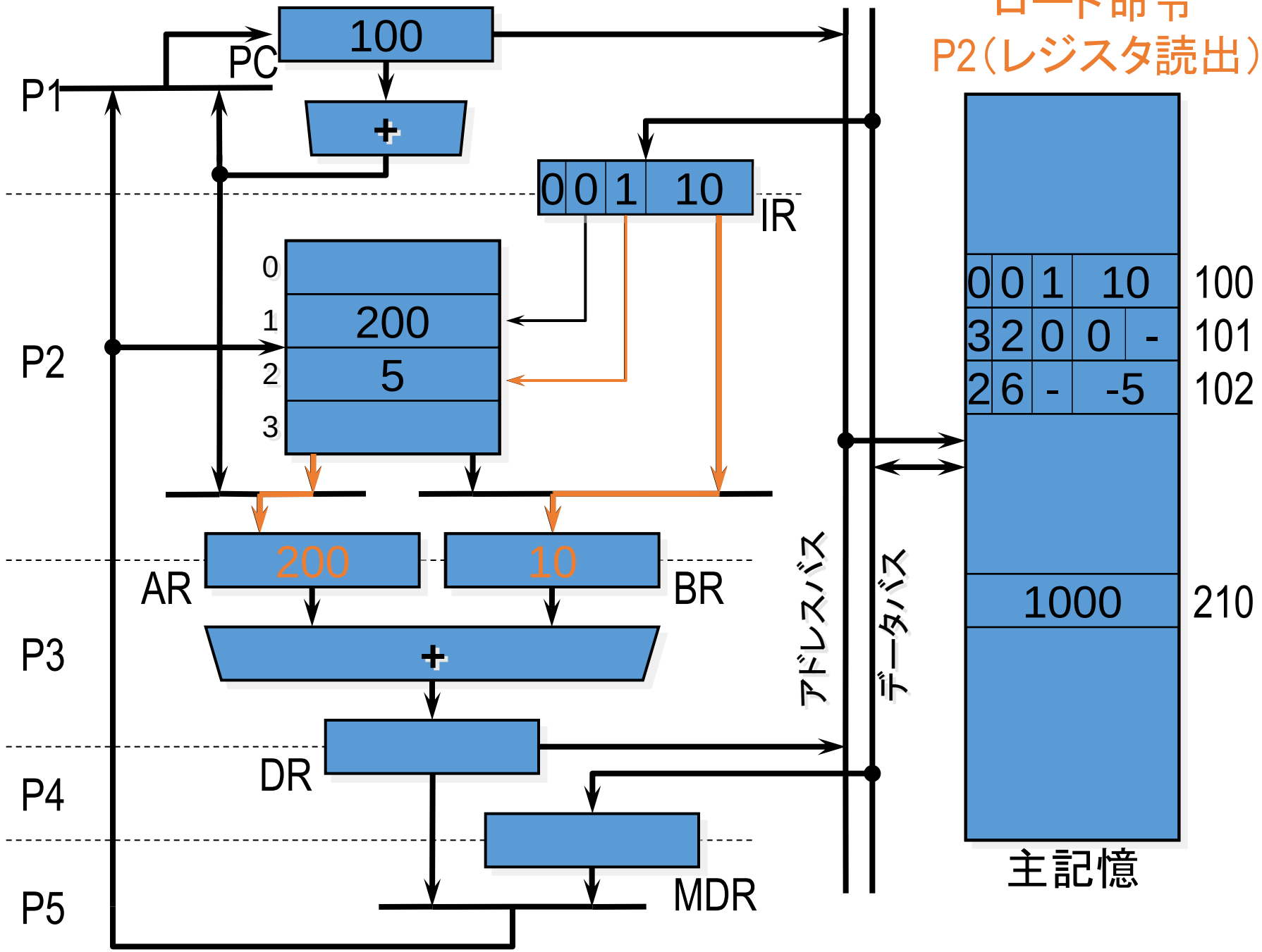
略記

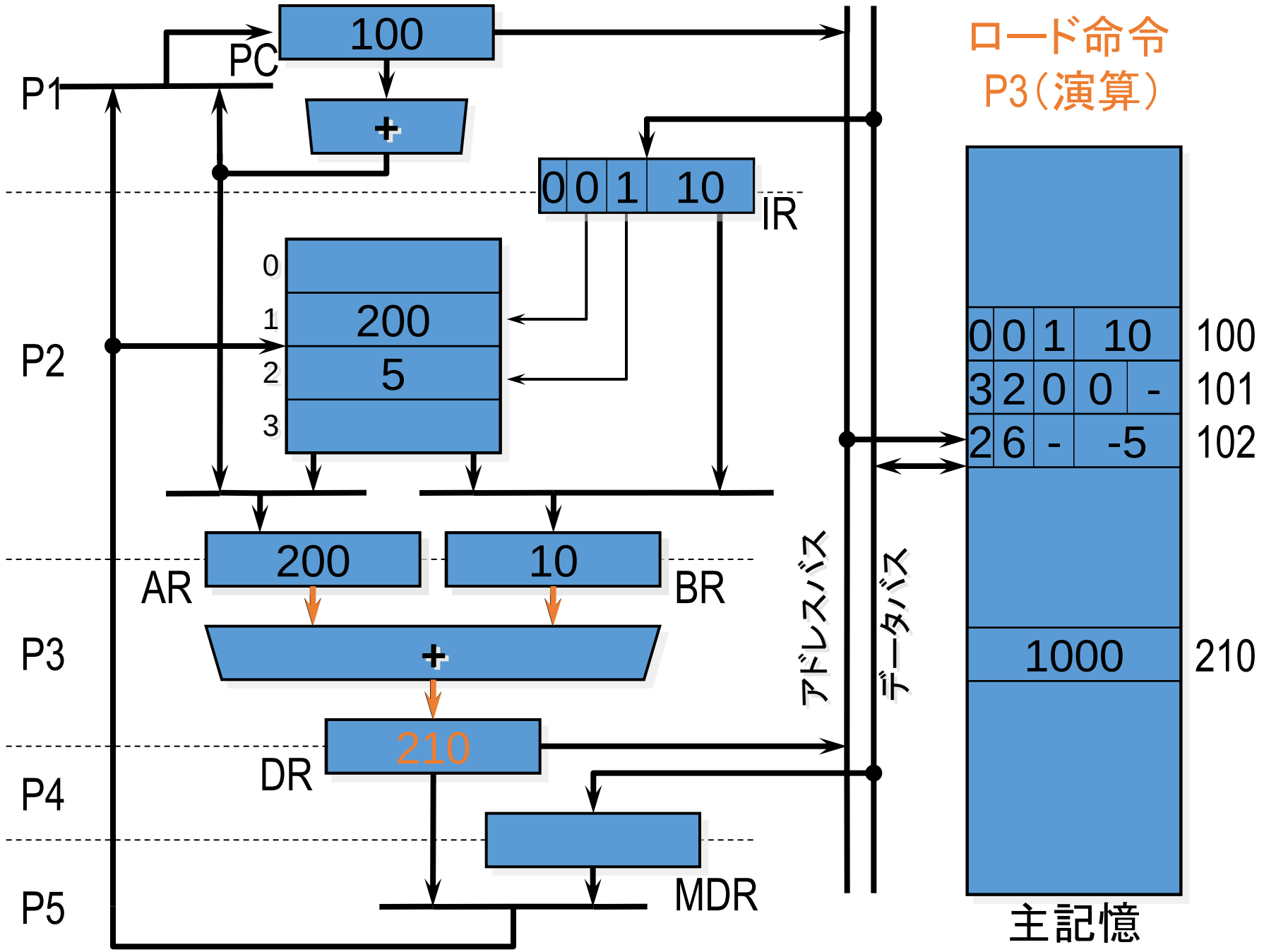
2	6	-	-5

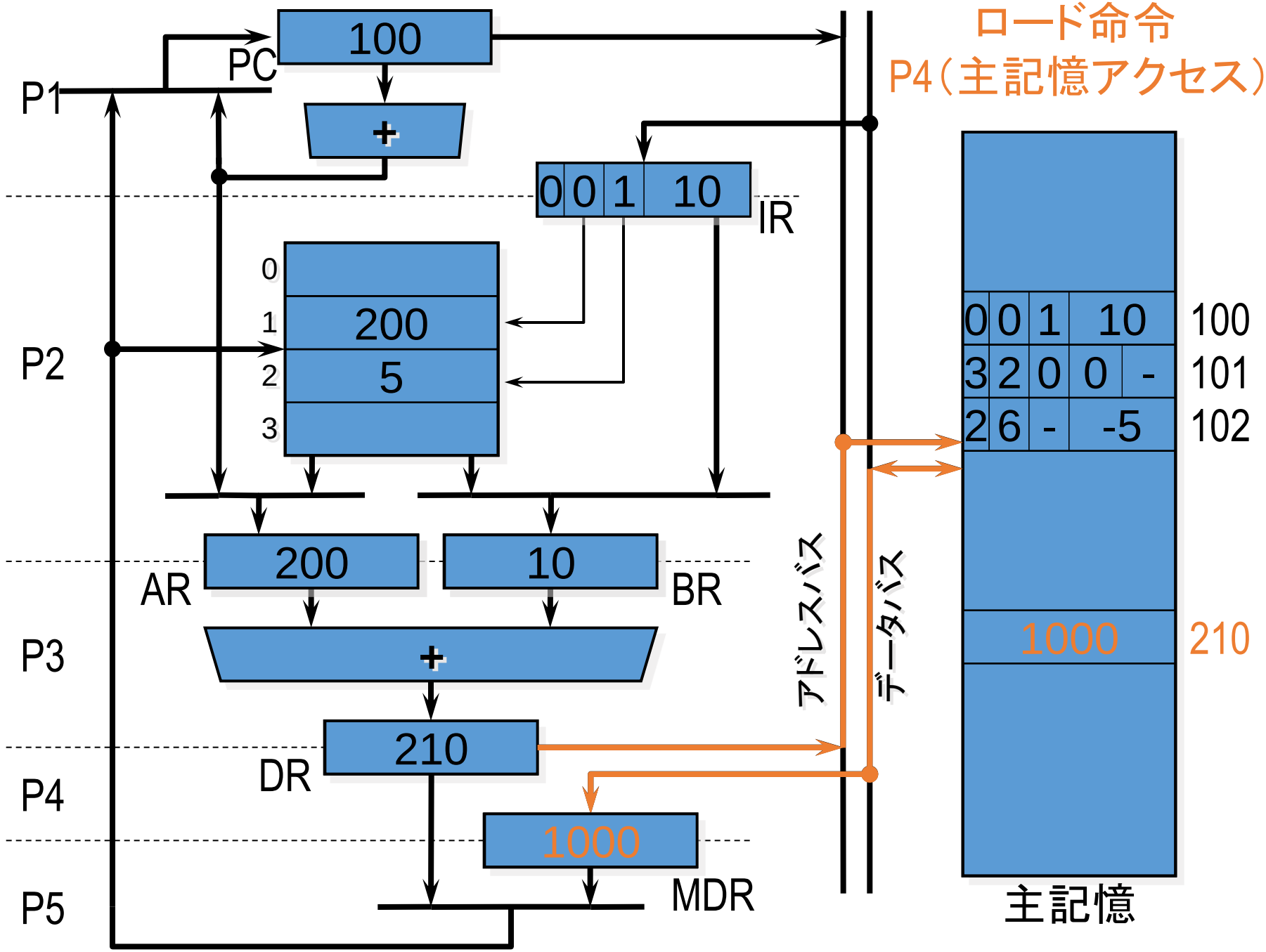


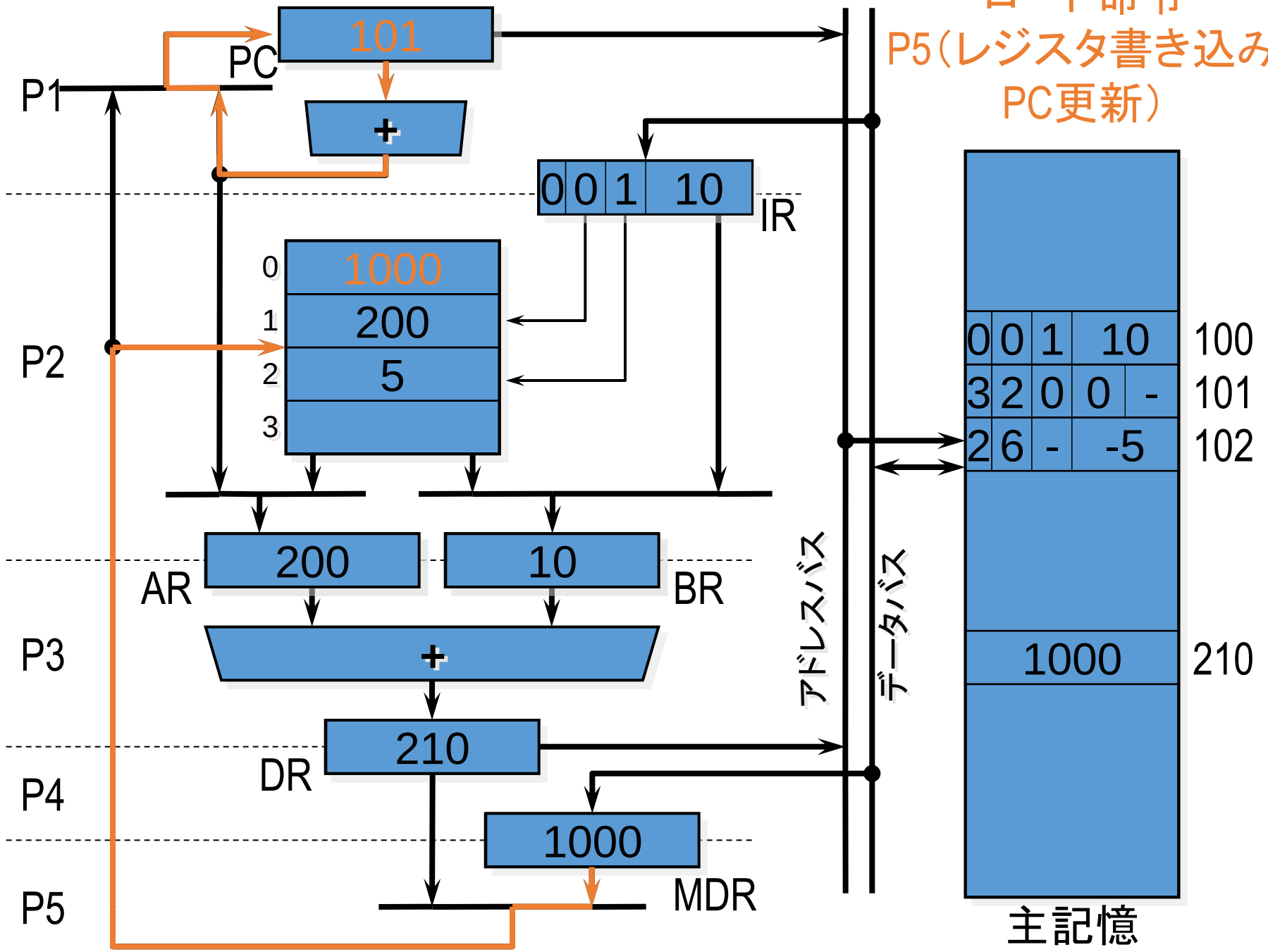

100
PC
P1
+
IR
0
1
200
2
5
3
P2
AR
BR
P3
+
DR
P4
MDR
P5
アドレスバス
データバス
0 0 1 10
100
3 2 0 0 -
101
2 6 - -5
102
1000
210
主記憶

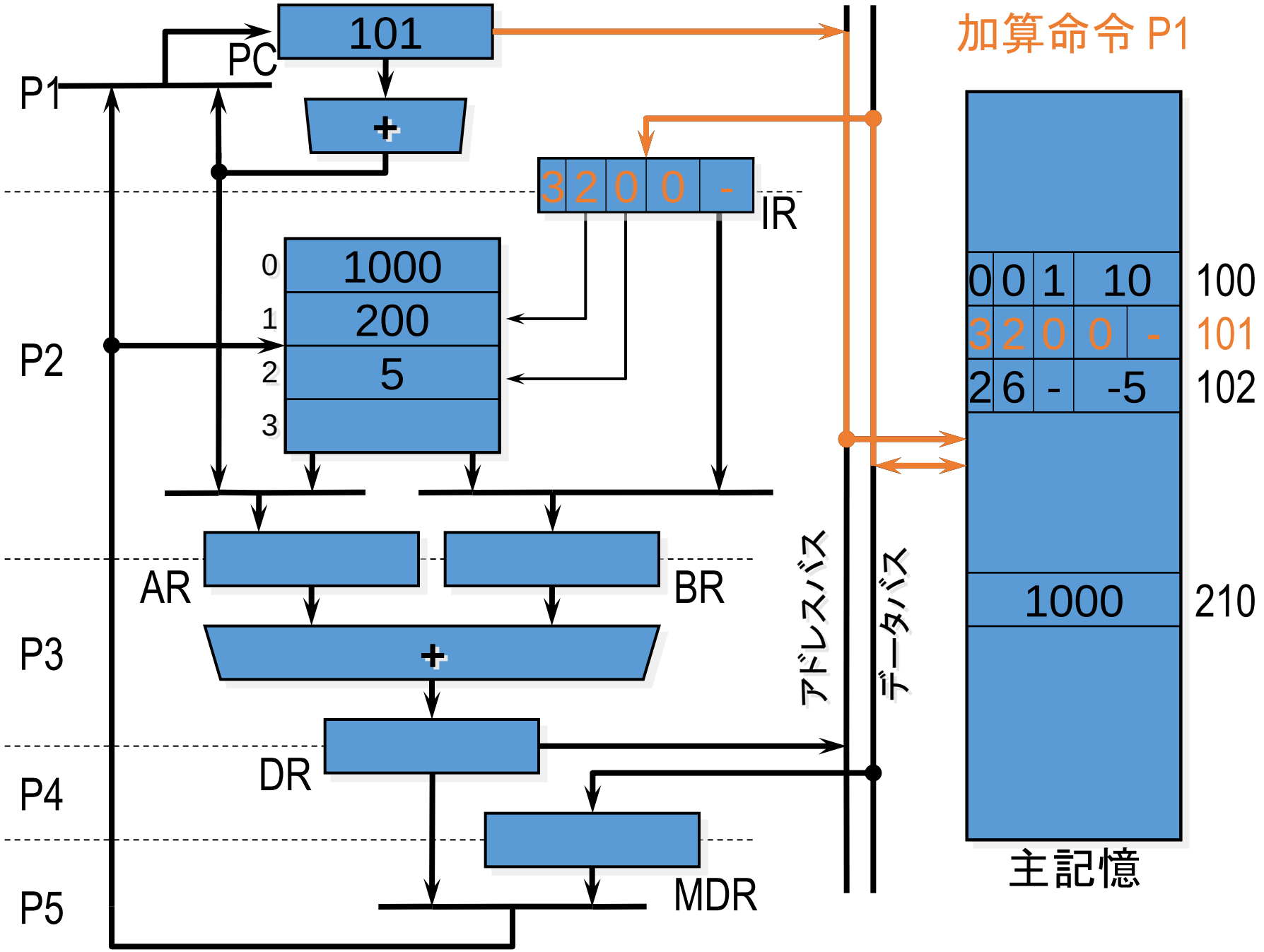

ロード命令
P1（命令フェッチ）
100
PC
P1
+
0 0 1 10
IR
0
1
200
2
5
3
P2
AR
BR
P3
+
DR
P4
MDR
P5
アドレスバス
データバス
0 0 1 10
100
3 2 0 0 -
101
2 6 - -5
102
1000
210
主記憶

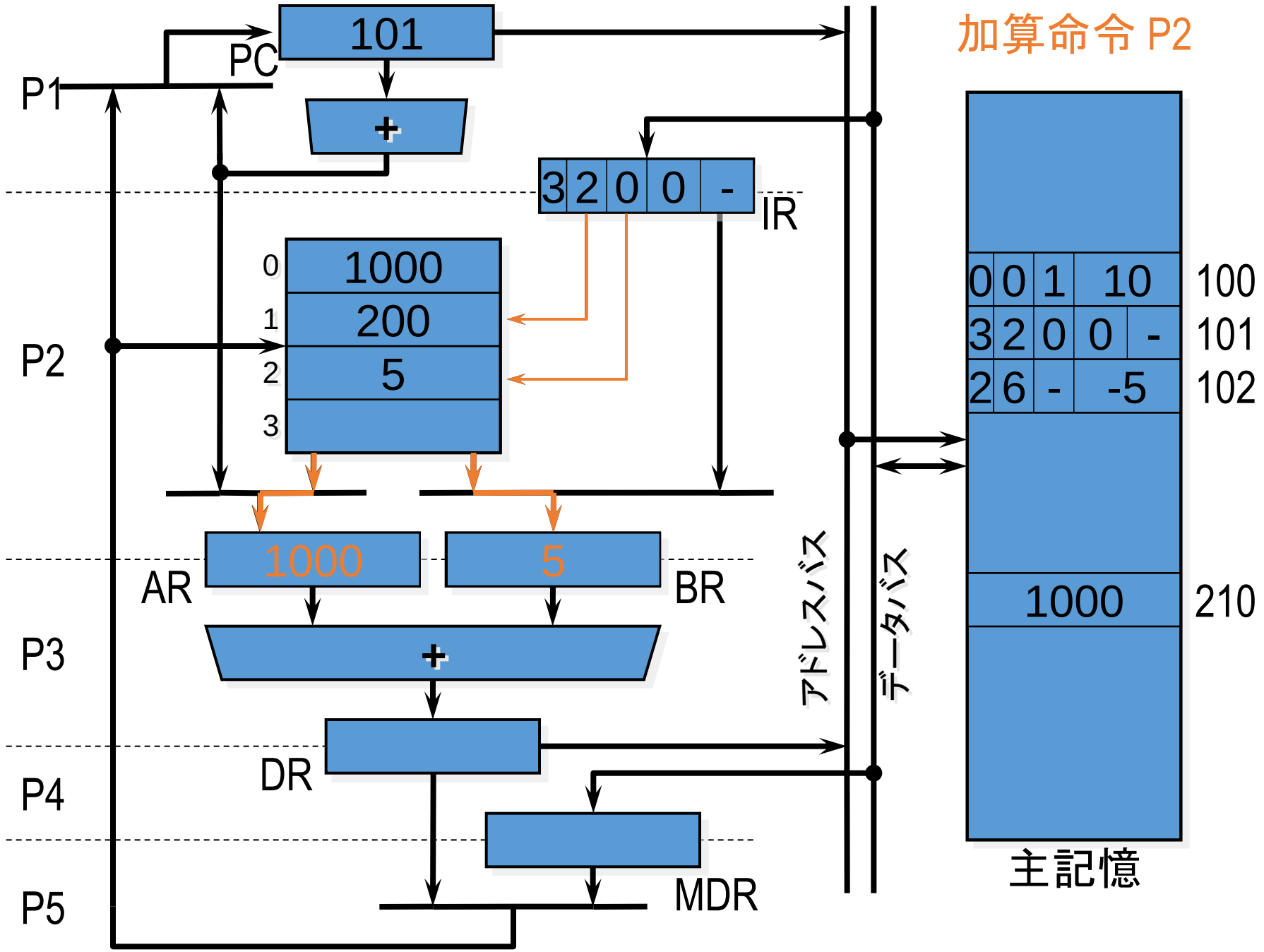

ロード命令
P2（レジスタ読出）
100
PC
P1
+
0 0 1 10
IR
0
1
200
2
5
3
P2
200
10
AR
BR
P3
+
DR
P4
MDR
P5
アドレスバス
データバス
0 0 1 10
100
3 2 0 0 -
101
2 6 - -5
102
1000
210
主記憶

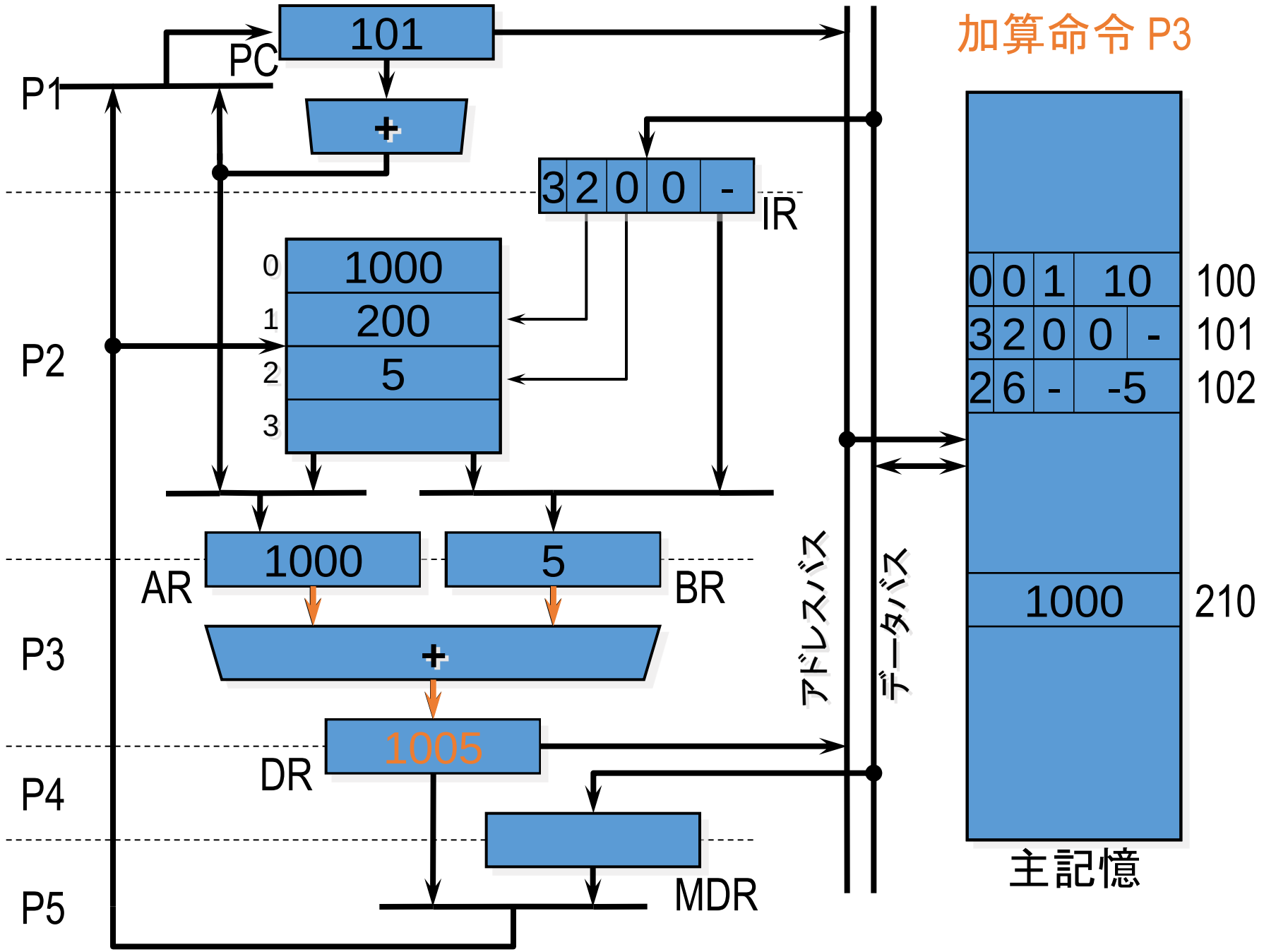

ロード命令
P3（演算）
100
PC
P1
+
0 0 1 10
IR
0
1
200
2
5
3
P2
200
10
AR
BR
P3
+
210
DR
P4
MDR
P5
アドレスバス
データバス
0 0 1 10
100
3 2 0 0 -
101
2 6 - -5
102
1000
210
主記憶

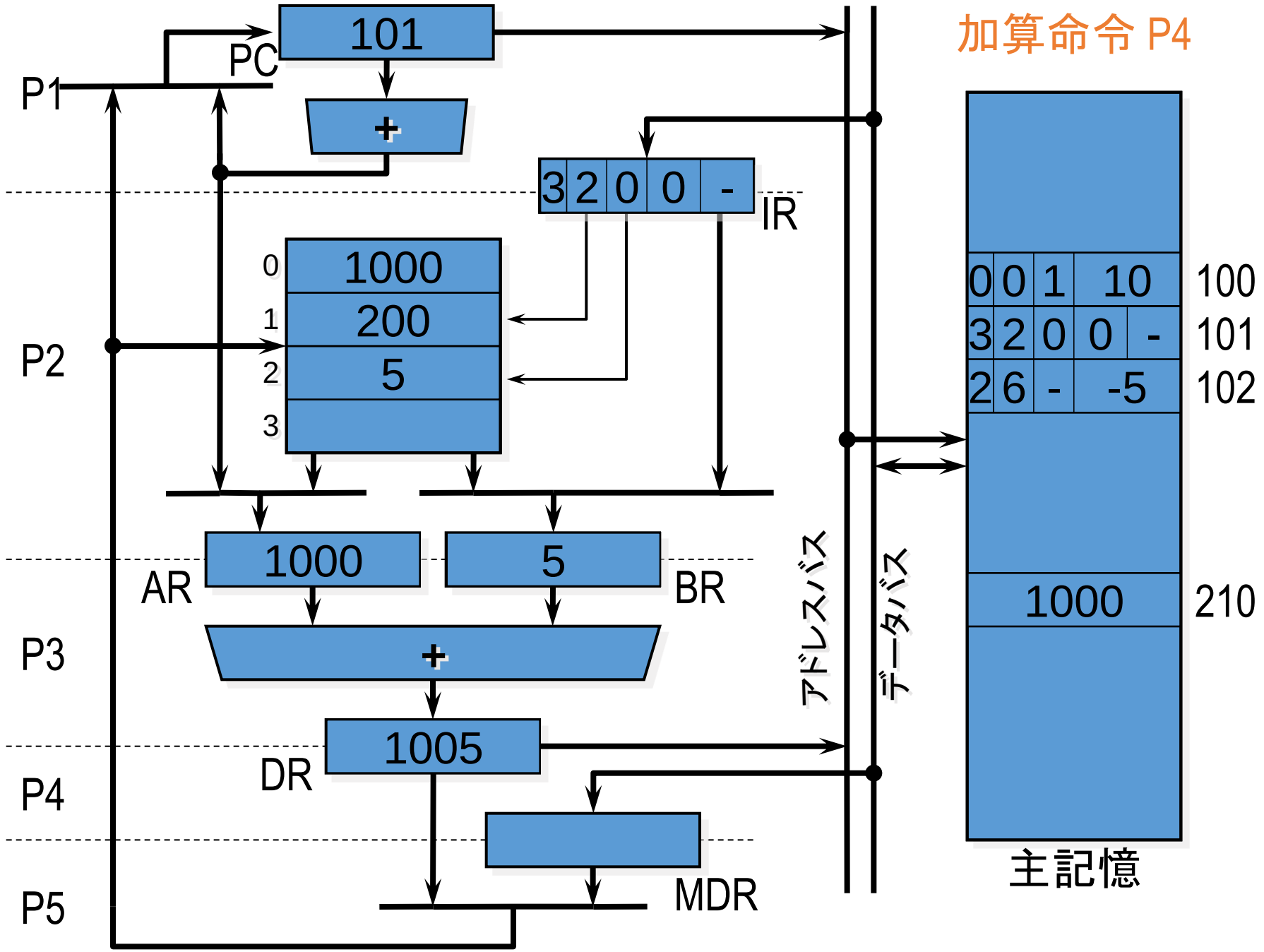

ロード命令
P4（主記憶アクセス）
100
PC
P1
+
0 0 1 10
IR
0
1
200
2
5
3
P2
200
10
AR
BR
P3
+
210
DR
P4
1000
MDR
P5
アドレスバス
データバス
0 0 1 10
100
3 2 0 0 -
101
2 6 - -5
102
1000
210
主記憶

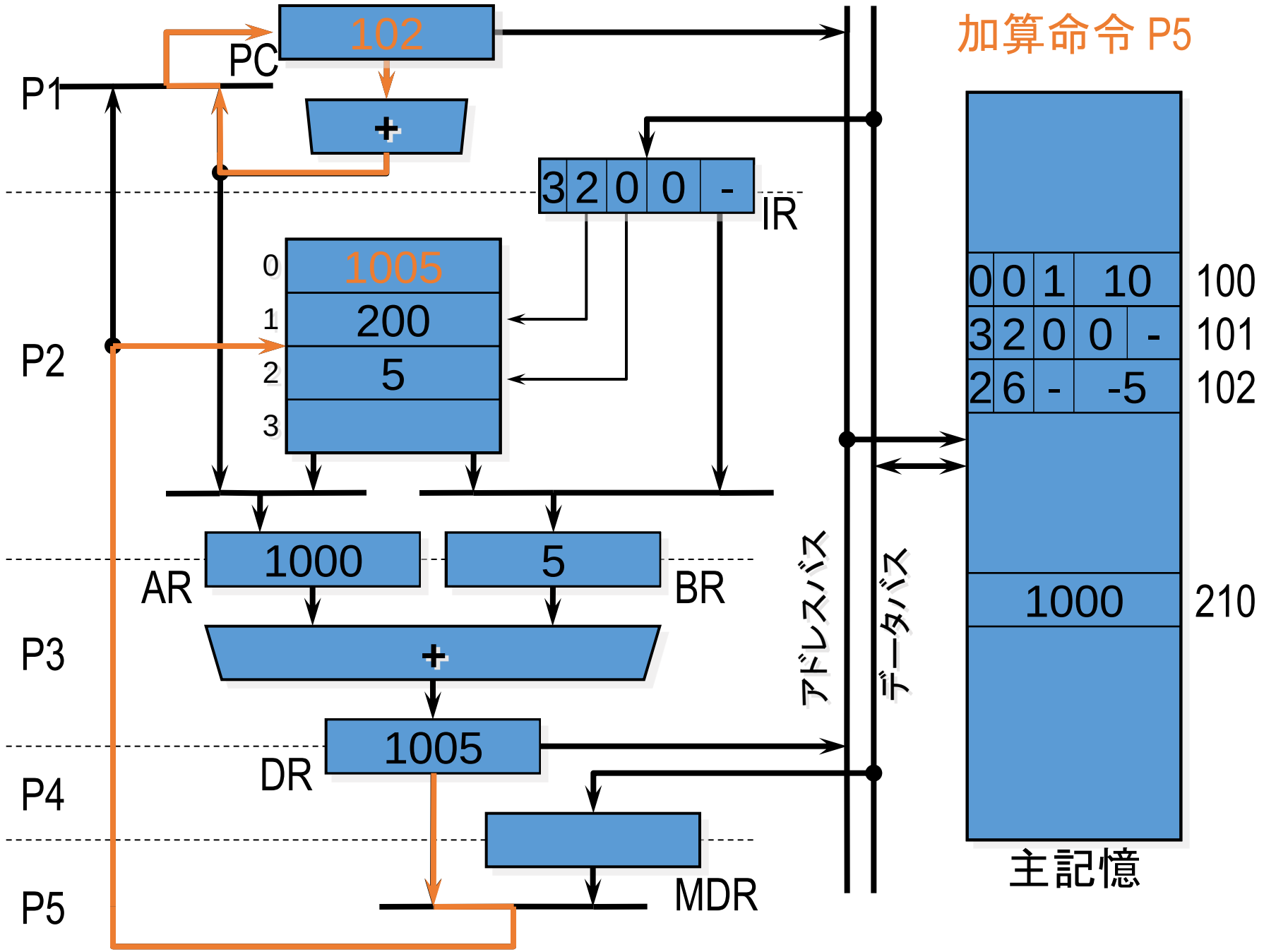

ロード命令
P5（レジスタ書き込み/
PC更新）
P1
P2
P3
P4
P5
PC
101
+
0 0 1 10
IR
0
1
2
3
1000
200
5
AR
200
10
BR
+
DR
210
1000
MDR
アドレスバス
データバス
0 0 1 10 100
3 2 0 0 - 101
2 6 - -5 102
1000 210
主記憶

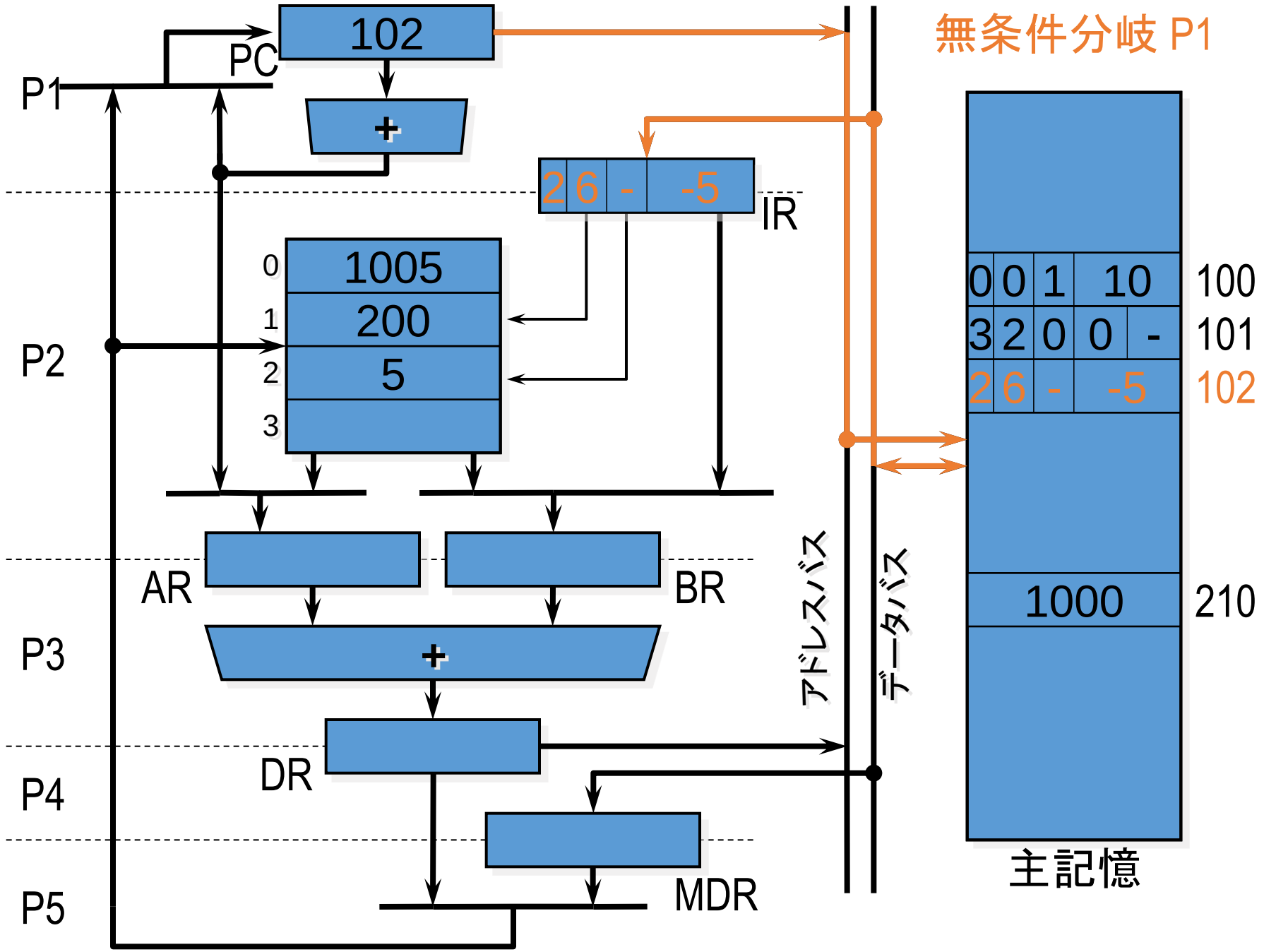

加算命令 P1
101
PC
P1
+
3200-
IR
0
1
2
3
1000
200
5
P2
AR
BR
P3
+
DR
P4
MDR
P5
アドレスバス
データバス
00110
320-
26--5
100
101
102
1000
210
主記憶

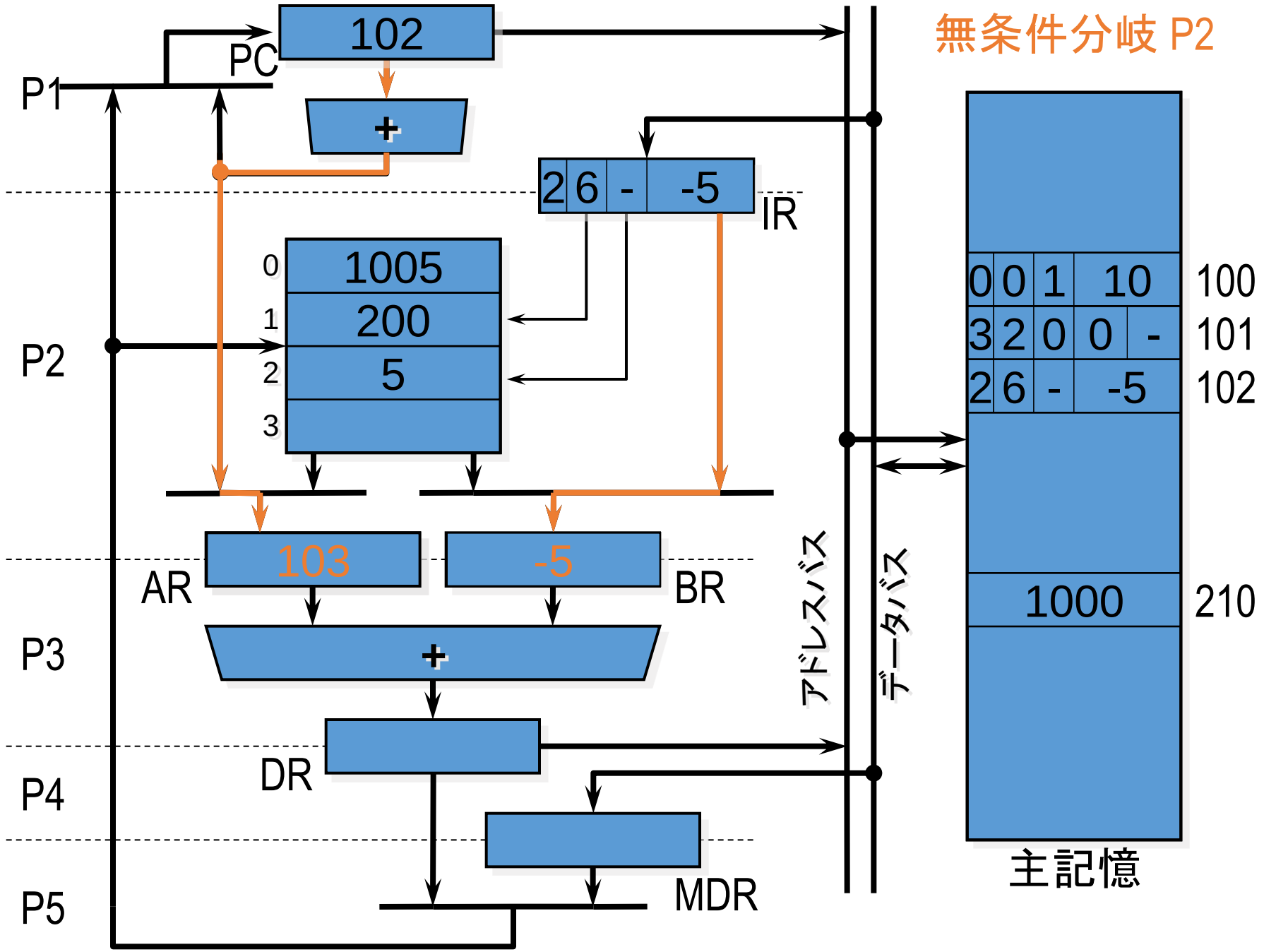

加算命令 P2
101
PC
P1
+
3200-
IR
0
1
2
3
1000
200
5
P2
1000
5
AR
BR
P3
+
DR
P4
MDR
P5
アドレスバス
データバス
0010
100
3200-
101
26-
-5
102
1000
210
主記憶

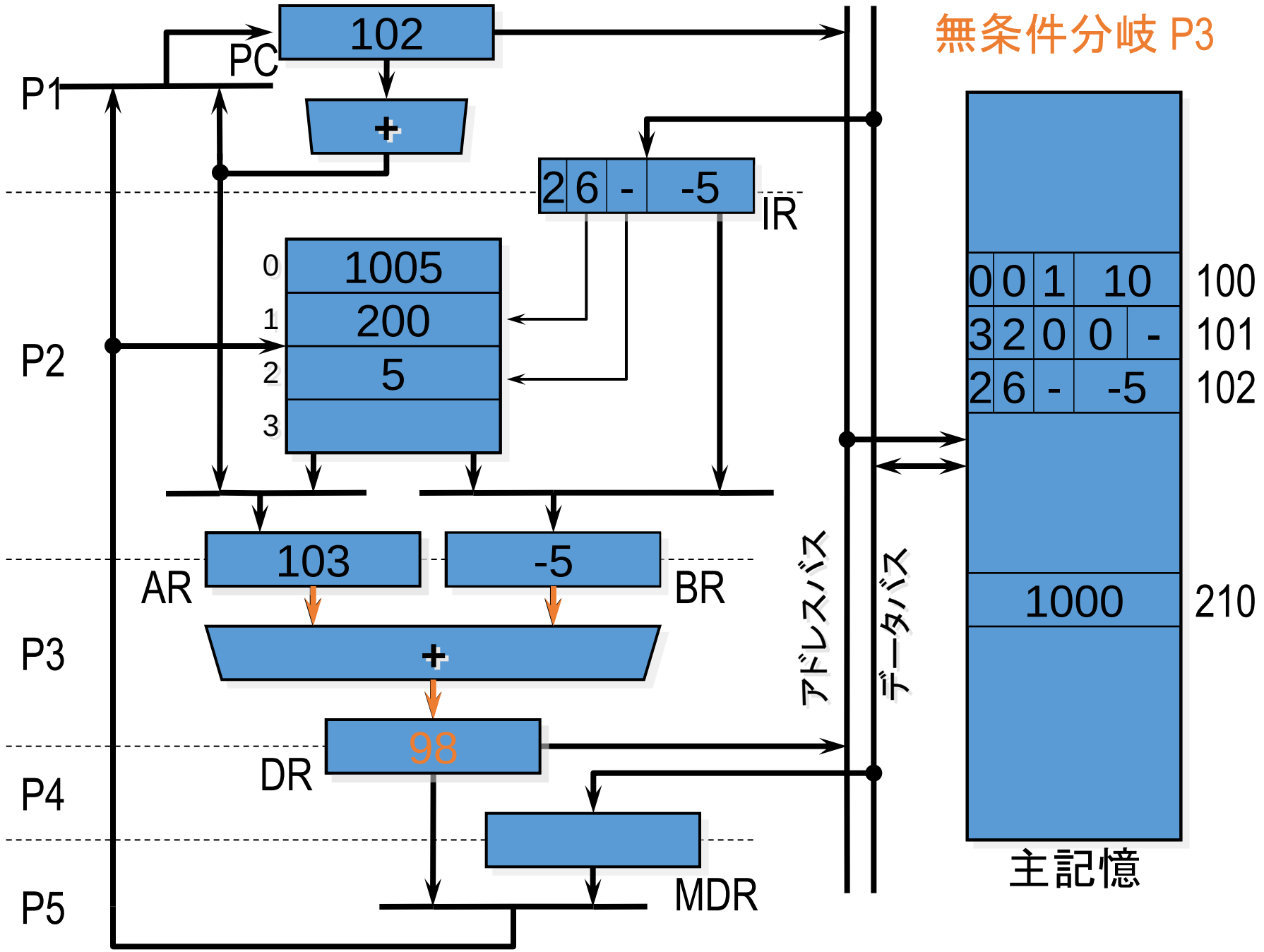

加算命令 P3
101
PC
P1
+
3200-
IR
0
1
2
3
1000
200
5
P2
1000
5
AR
BR
P3
+
1005
DR
P4
MDR
P5
アドレスバス
データバス
00110
32 0 0 -
26 - -5
100
101
102
1000
210
主記憶

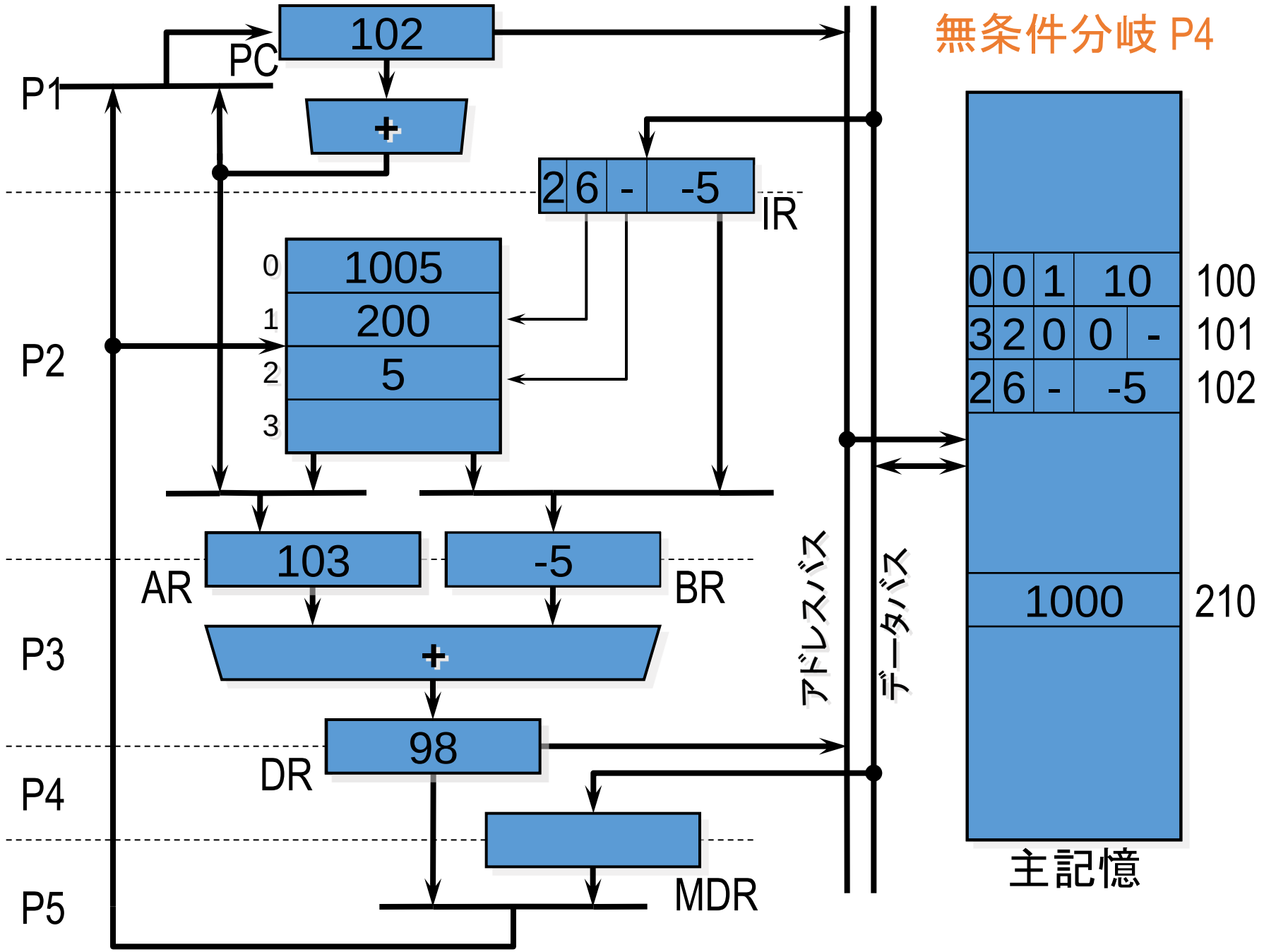

加算命令 P4
101
PC
P1
+
3200-
IR
0
1
2
3
1000
200
5
P2
1000
5
AR
BR
P3
+
1005
DR
P4
MDR
P5
アドレスバス
データバス
0010
100
3200-
101
26-
-5
102
1000
210
主記憶

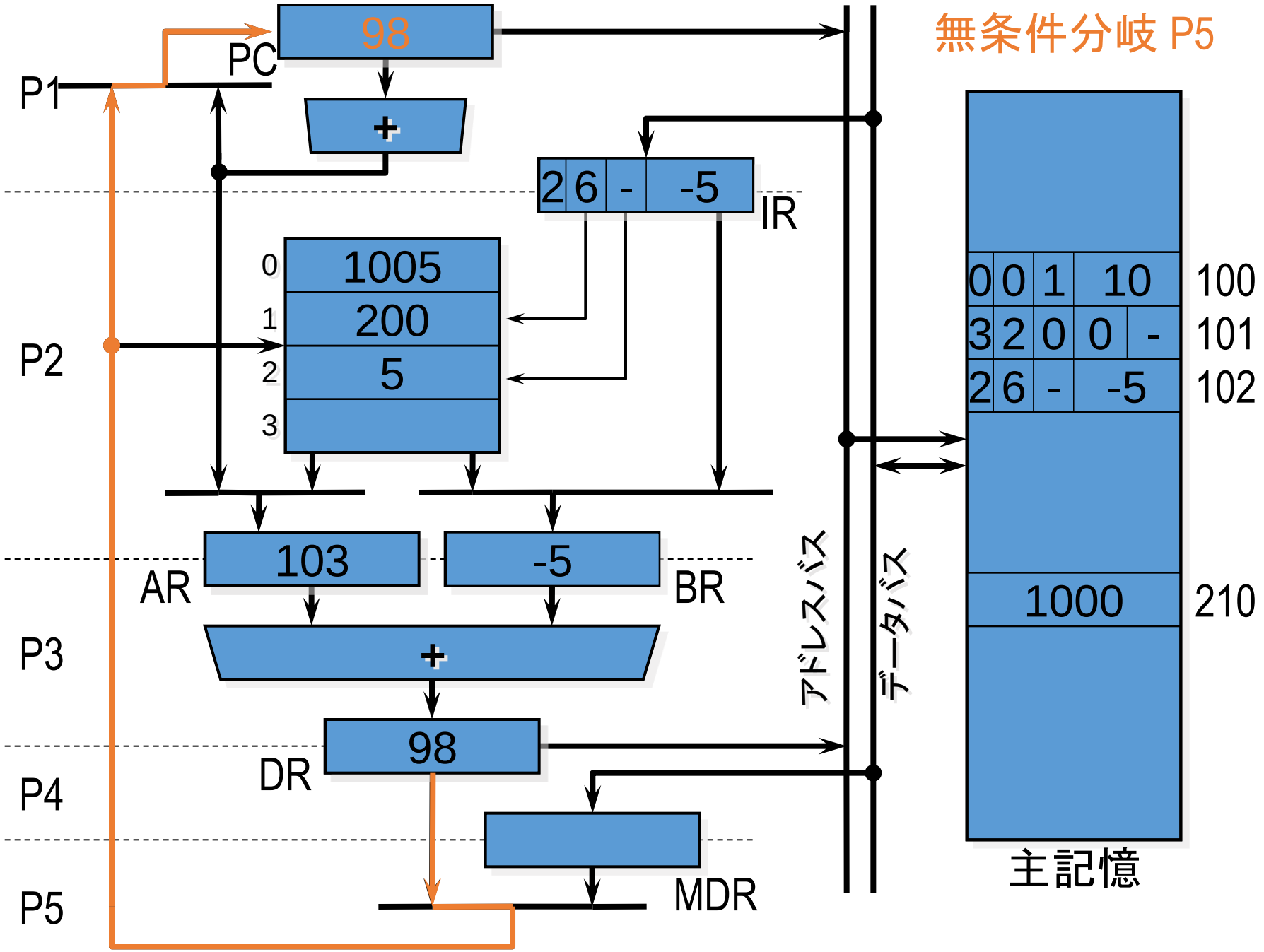

加算命令 P5
102
PC
P1
+
3200-
IR
0
1005
1
200
2
5
3
P2
1000
5
AR
BR
P3
+
1005
DR
P4
MDR
P5
アドレスバス
データバス
0010
10
100
3200-
101
26-
-5
102
1000
210
主記憶


無条件分岐 P1
102
PC
P1
+
26
-
-5
IR
0
1
2
3
1005
200
5
P2
AR
BR
P3
+
DR
P4
MDR
P5
アドレスバス
データバス
00
1
10
100
32
0
0
-
101
26
-
-5
102
1000
210
主記憶


無条件分岐 P2
102
PC
P1
+
2 6 - -5
IR
0
1
2
3
1005
200
5
P2
103
-5
AR
BR
P3
+
DR
P4
MDR
P5
アドレスバス
データバス
0 0 1 10
100
3 2 0 0 -
101
2 6 - -5
102
1000
210
主記憶


無条件分岐 P3
102
PC
P1
+
2 6 - -5
IR
0
1
2
3
1005
200
5
P2
103
-5
AR
BR
P3
+
98
DR
P4
MDR
P5
アドレスバス
データバス
0 0 1 10
3 2 0 0 -
2 6 - -5
1000
100
101
102
210
主記憶


無条件分岐 P4
102
PC
P1
+
2 6 - -5
IR
0
1
2
3
1005
200
5
P2
103
-5
AR
BR
P3
+
98
DR
P4
MDR
P5
アドレスバス
データバス
0 0 1 10
3 2 0 0 -
2 6 - -5
1000
100
101
102
210
主記憶


無条件分岐 P5
98
PC
P1
+
2 6 - -5
IR
0
1
2
3
1005
200
5
P2
0 0 1 10
3 2 0 0 -
2 6 - -5
100
101
102
1000
210
103
-5
AR
BR
P3
+
98
DR
P4
MDR
P5
アドレスバス
データバス
主記憶

課題とレポート

課題

- SIMPLE/Bから**何らかの拡張**を行って、拡張前と**比較評価**する
  - 拡張の例：パイプライン化、命令の追加、スーパースカラ実行
  - 拡張によってプログラムの実行がどれだけ高速化したか、追加で必要になったハードウェアなどを評価
    - 最高クロック周波数、実行命令数、実行サイクル数
    - ゲート数（LUT数）
- デモンストレーション（後述）で披露する**応用プログラム**の作成
  - 独自プロセッサ上で動作する応用プログラムを作成する
  - 過去の例：音楽再生、テトリス、フィボナッチ数列の計算、ソート etc…

レポートとデモンストレーション

- 中間デモ 5/ 2 (金)
  - 何らかの命令が動作しているところを見せる。
- 中間レポート 5/2(金)
  - 独自プロセッサの概要をまとめた報告書
  - そのプロセッサを実現するための命令セット・アーキテクチャ、構造
  - 各コンポーネントの仕様書
  - 実施状況報告
- 最終デモ 5/23(金)
  - 独自プロセッサ上で実行可能な応用プログラムを作成。それを使ったプロセッサの特徴の説明
    - 「SIMPLE/Bに比べてこんな点がすぐれている」というセールストーク
    - プロセッサの特徴を活かした応用プログラムの実行
- 最終レポート 5/30(金)
  - 最終成果物のユーザーズマニュアル
  - SIMPLE/B基本仕様からの拡張および性能評価
  - 各コンポーネントの仕様書（最終版）、性能評価
  - 考察、感想

- 単に動くだけではなく、性能、回路規模の最適化を目指す
- デモは 1 グループ 5 分ほど。推しポイントをメインにプレゼン

ソート速度コンテスト

- データをソートする時間を競うコンテスト
  - データ
    - 16bitの符号付整数1024個
    - ランダム、昇順ソート済み、降順ソート済みの3種類
  - 時間の定義：完了までのサイクル数×クロック周波数
  - 3種類のデータ各々の処理時間の平均値
- 過去の先輩の記録も記載。ぜひ参加して、歴代最高記録を狙ってください
  http://isle3hw.kuis.kyoto-u.ac.jp/contest/index.html
- 随時参加できます。スタッフに声をかけてください。

実験の進め方・Tips

モジュール構成と分担を決める

- どのようなモジュール構成にするか
  - プロセッサ全体をどう分割するか
  - Verilog HDLのモジュール単位？機能のブロック単位？
  - 各レジスタはどの単位に属するか
- どう分担を分けるか
  - 制御系とデータパス系？
  - サブデザインとトップデザイン＆インタフェース？
  - 基本機能と拡張機能？
  - 同じ機能ブロックの違うバージョンをそれぞれ設計？

進め方、スケジュールを決める

- 進め方
  - 部品から作るか、トップデザイン（部品はダミー）をまず用意するか。両方から攻めるのもあり。
  - 最も単純な機能や構成から始めるか、機能拡張に備えた構成をまず考えるか
- スケジューリング
  - 中間レポート「何らかの命令が動作」の実現時期と機能をどう設定するか（中間まで約3週間。意外と短い）
  - 最終成果物の仕様は中間レポートまでに決める。修正は後でもできる。
- 予定通りには進まない
  - 工程が後戻りすることもある。スケジュールは余裕を持って立てておくこと。

検証（デバッグ）環境を整える

- テストベンチを作り込む
  - 必要な入力信号の変化をすべてテストベンチに記載しておくことで、シミュレーションを立ち上げるだけで、動作確認ができるようになる。
- 実機での検証環境の構築
  - 回路規模が大きくなってくると、シミュレーションでのデバッグは大変。
  - ボード上のLEDや7セグLEDを利用して、プロセッサの内部信号（レジスタや制御信号の値）を表示できるようにする。