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

2023/04 rev

この講義の内容

- この講義は実験３ハードウェアの課題とSIMPLEの仕様の簡単な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。
- SIMPLEの仕様の詳細は実験HPのSIMPLE設計資料にあります。

実験３ハードウェアの内容と目的

- 内容
  - マイクロプロセッサの設計（方式設計、論理設計）
    - SIMPLE/Bを拡張し独自のプロセッサを設計する
  - FPGA上で応用プログラムを動作
- 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プロセッサアーキテクチャ

アーキテクチャの説明

- アーキテクチャ
  - コンピュータ全体の構成
    - プロセッサ、メモリ、I/Oなど
  - 主記憶とレジスタの構成ここに含む
- 命令セットアーキテクチャ
  - 命令の構成
  - 例：x86, Intel 64など
- マイクロアーキテクチャ
  - 回路レベルでの実装
  - 例：skylake、zenなど

SIMPLE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
  - 8本の汎用レジスタ
  - 16bit×64K語の主記憶
  - ロード/ストアアーキテクチャ
  - 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

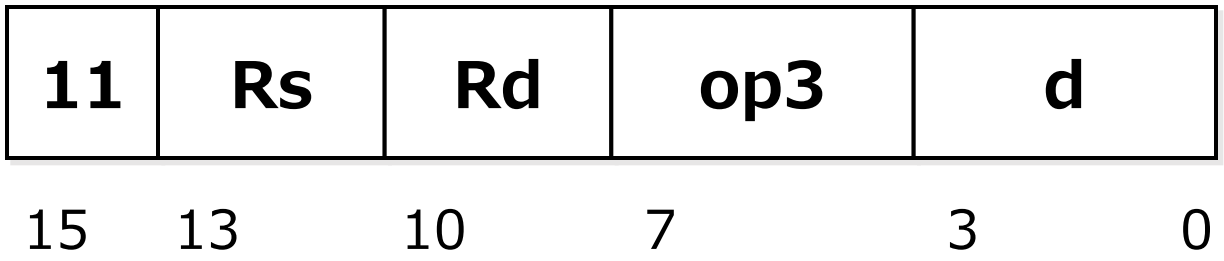
1. 主記憶
   - 16bit×64K語　（語アドレス方式）
   - ただし、実験で使用するFPGAで確保できる最大サイズは約33K語
2. 汎用レジスタ
   - 16bit×8語
3. プログラムカウンタ (PC)
   - 16bit
4. 条件コード
   - S　サイン
   - Z　ゼロ
   - C　キャリー
   - V　オーバーフロー

命令セット

1. 演算命令
   - 算術論理演算命令
   - シフト命令
2. ロード／ストア命令
3. 分岐命令
   - 無条件分岐命令
   - 条件分岐命令
4. その他
   - 入出力命令
   - 停止命令

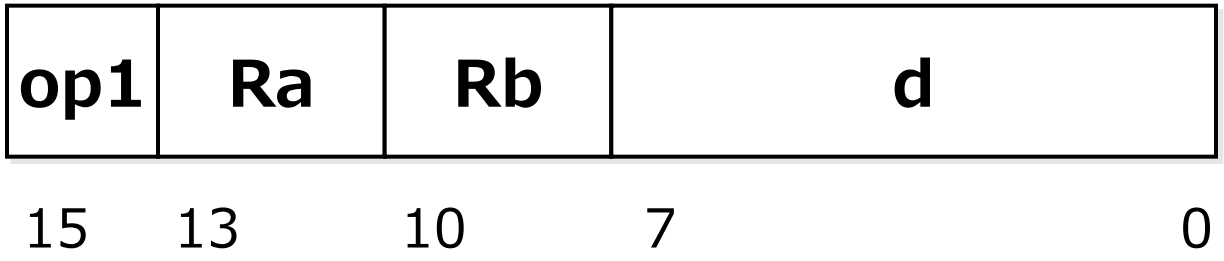
演算命令

- 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



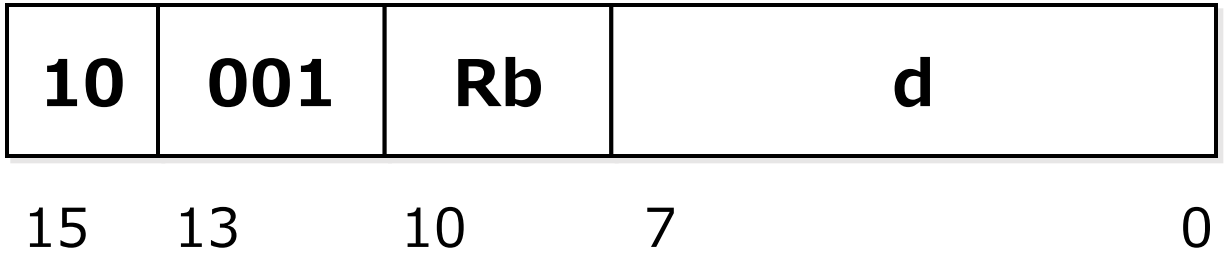
ロード／ストア命令(1)

- 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]**



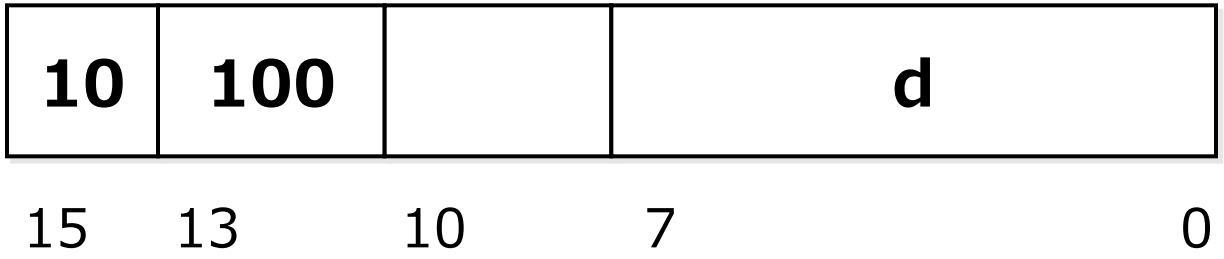
ロード／ストア命令(2)

- 即値ロード命令
  - **r[Rb] = sign_ext(d)**
  - 即値ロード命令２つとシフト命令で任意の１６bitの値をレジスタ格納できる



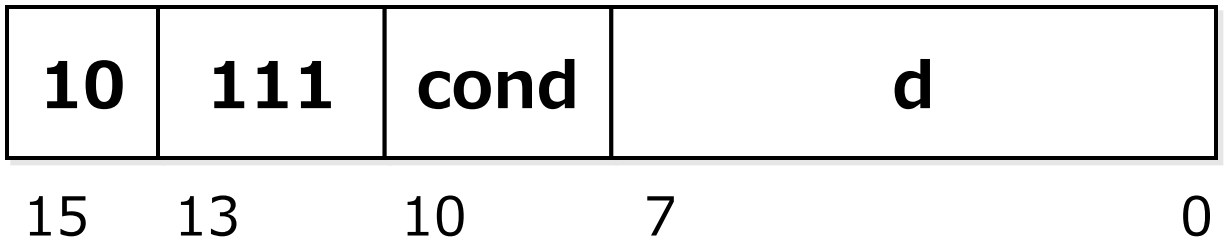
分岐命令(1)

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



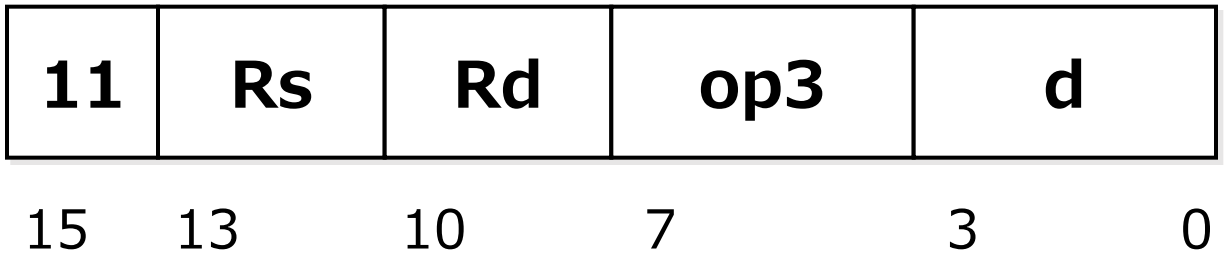
分岐命令(2)

- 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



その他の命令

- 停止命令**(op3: 1111)**
- 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



基本的な実装　SIMPLE/B

- 次スライドに示すように演算器／レジスタ／データパスを配置
- 5つのフェーズを逐次活性化：実験2の順序回路と同じ
  - 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
- フェーズの活性化：制御部が担当
  - (フェーズへ入力されるデータを保持するレジスタを更新)
  - フェーズ内のセレクタ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

命令実行の例

```
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。(ジャンプ)

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

00	Ra (000)	Rb (001)	d (00001010)

15 13 10 7 0

- 加算命令： プログラムカウンタ101
  - ADD R0, R2

略記 3 2 0 0 -

11	Rs (010)	Rd (000)	op3 (0000)	d

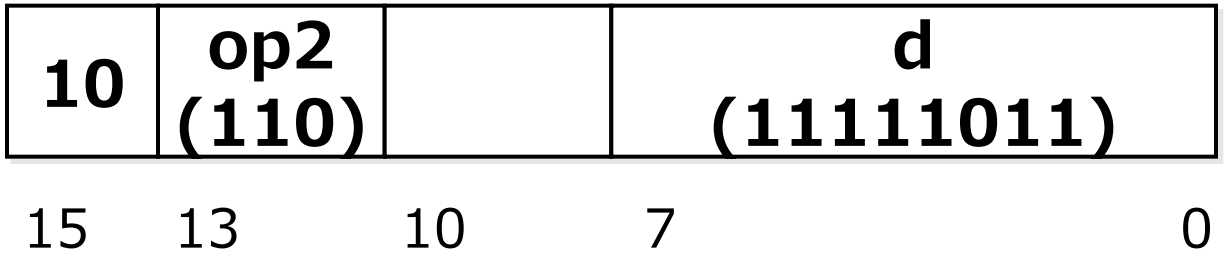
15 13 10 7 3 0

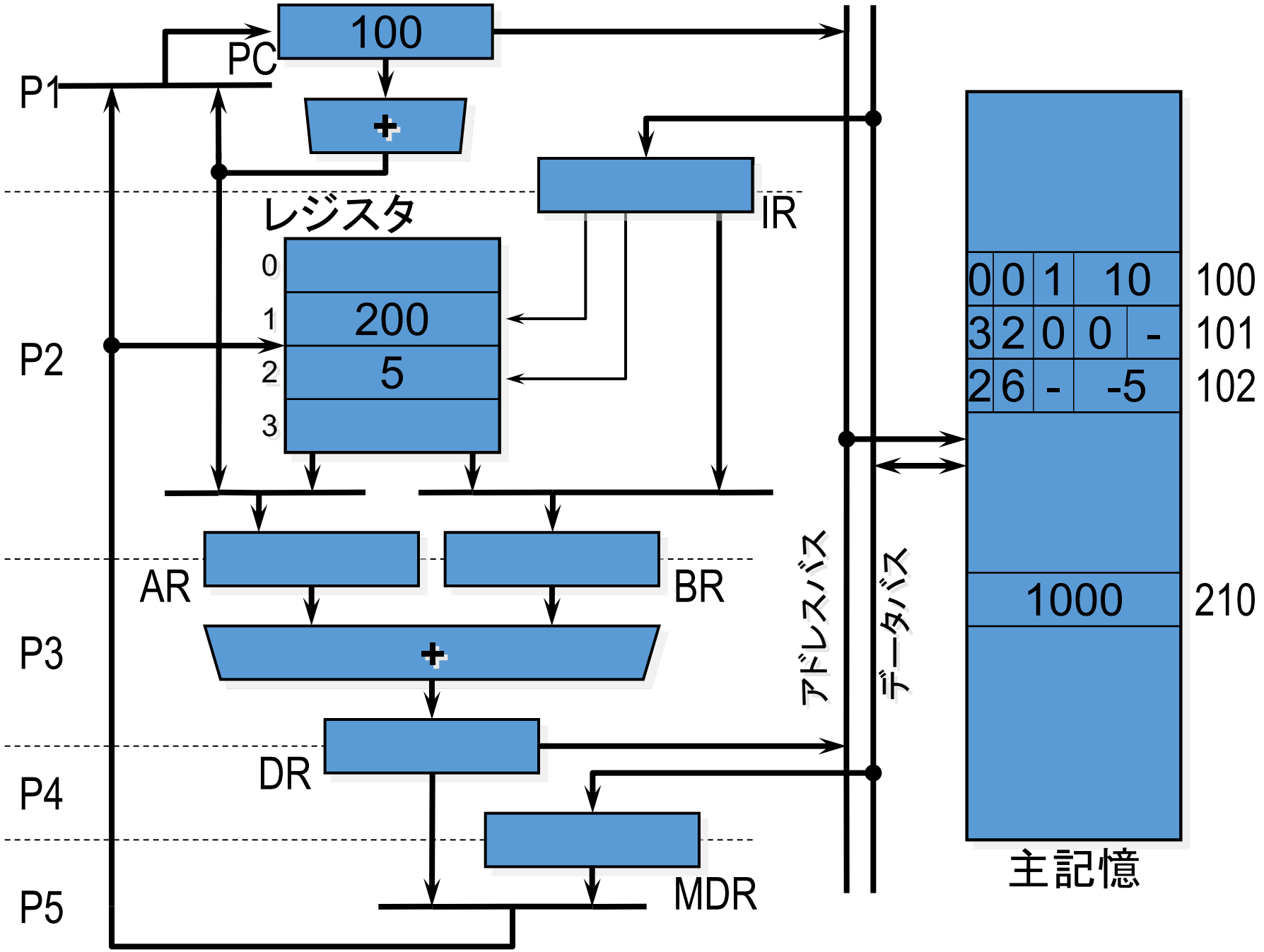
命令実行の例

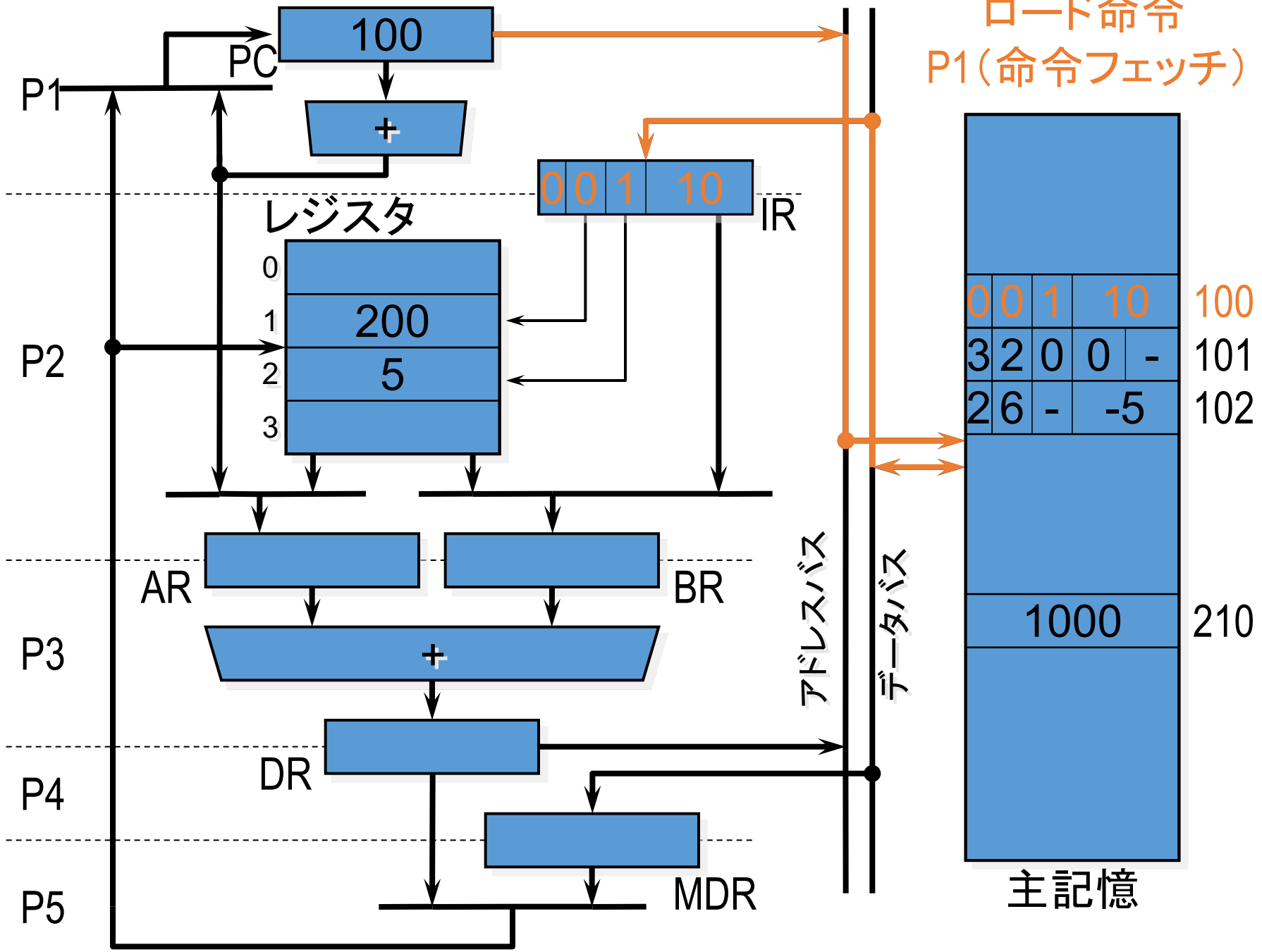
- 無条件分岐命令：プログラムカウンタ102
  - B -5

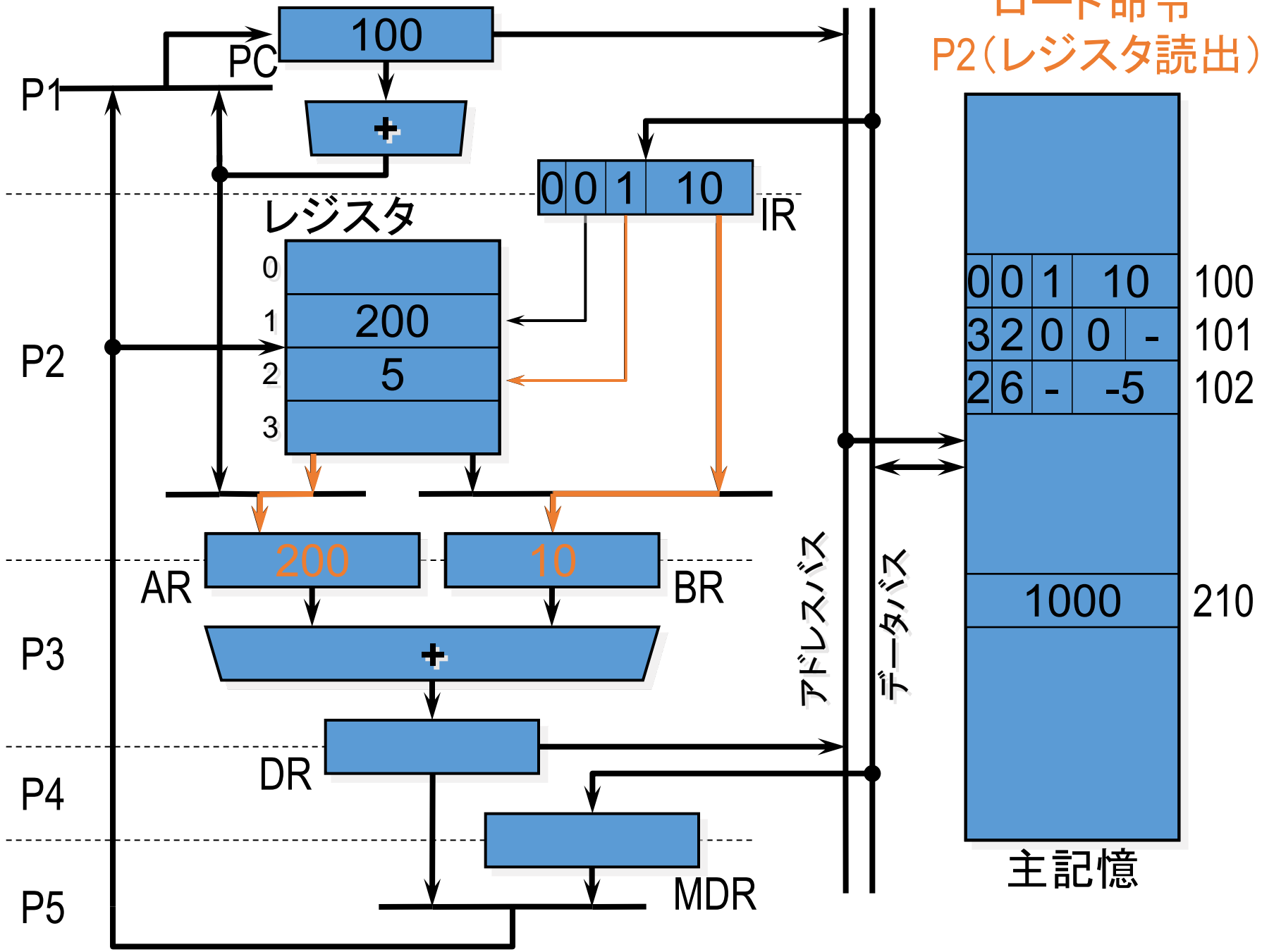
略記

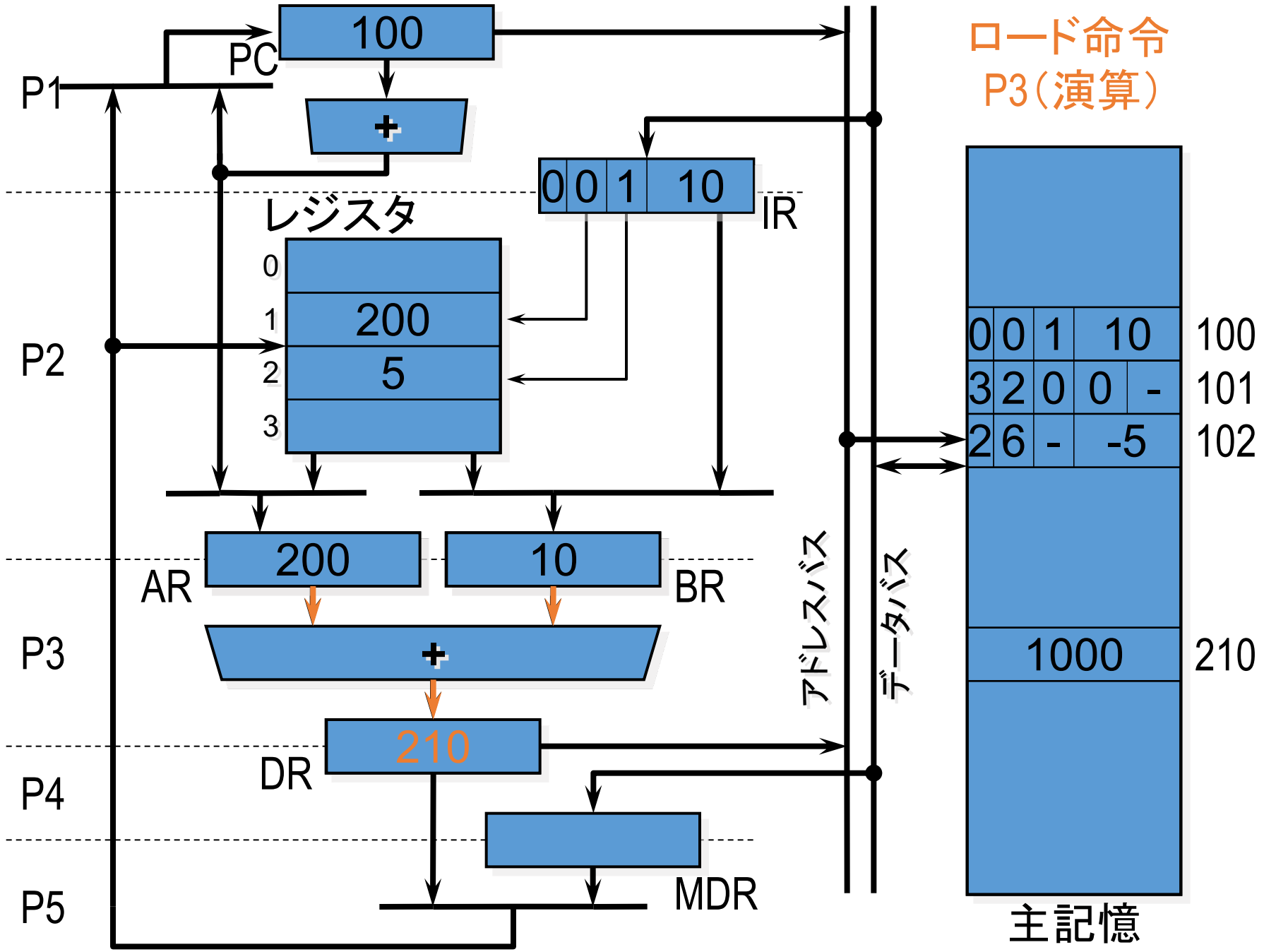
2	6	-	-5

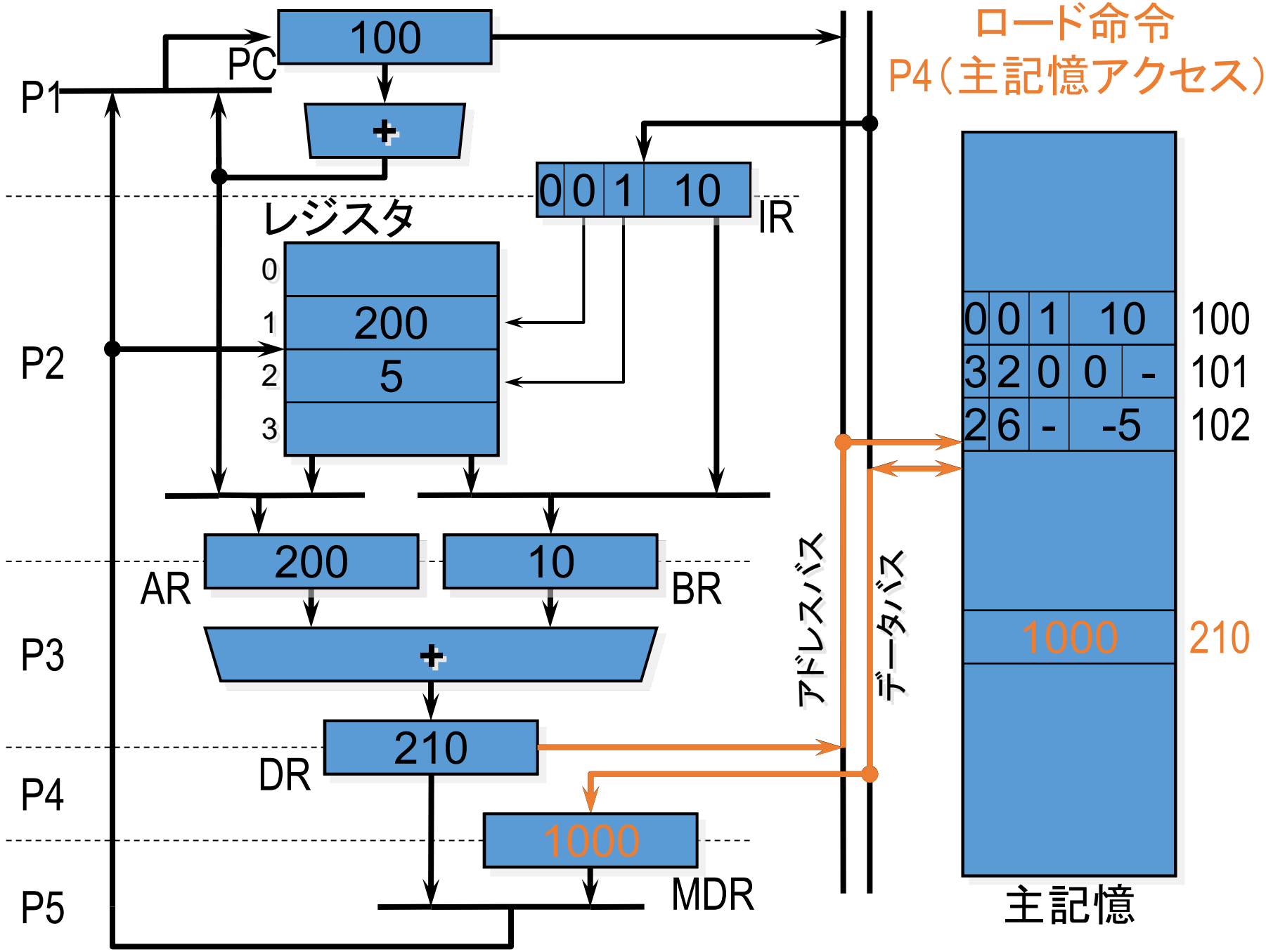


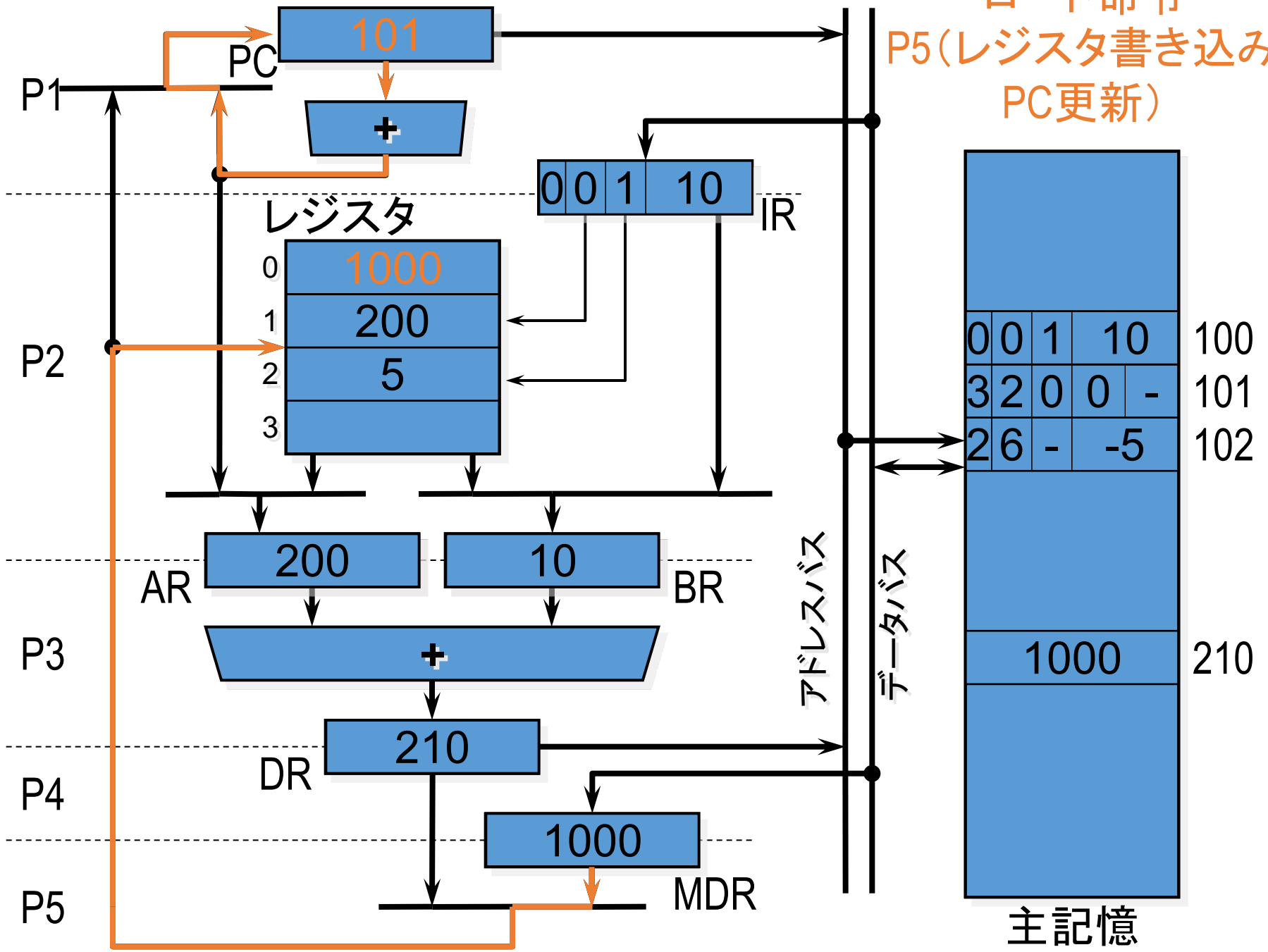

100
PC
P1
+
IR
レジスタ
0
1
200
2
5
3
P2
AR
BR
P3
+
DR
P4
MDR
P5
アドレスバス
データバス
0 0 1 10
100
3 2 0 0 -
101
2 6 - -5
102
1000
210
主記憶

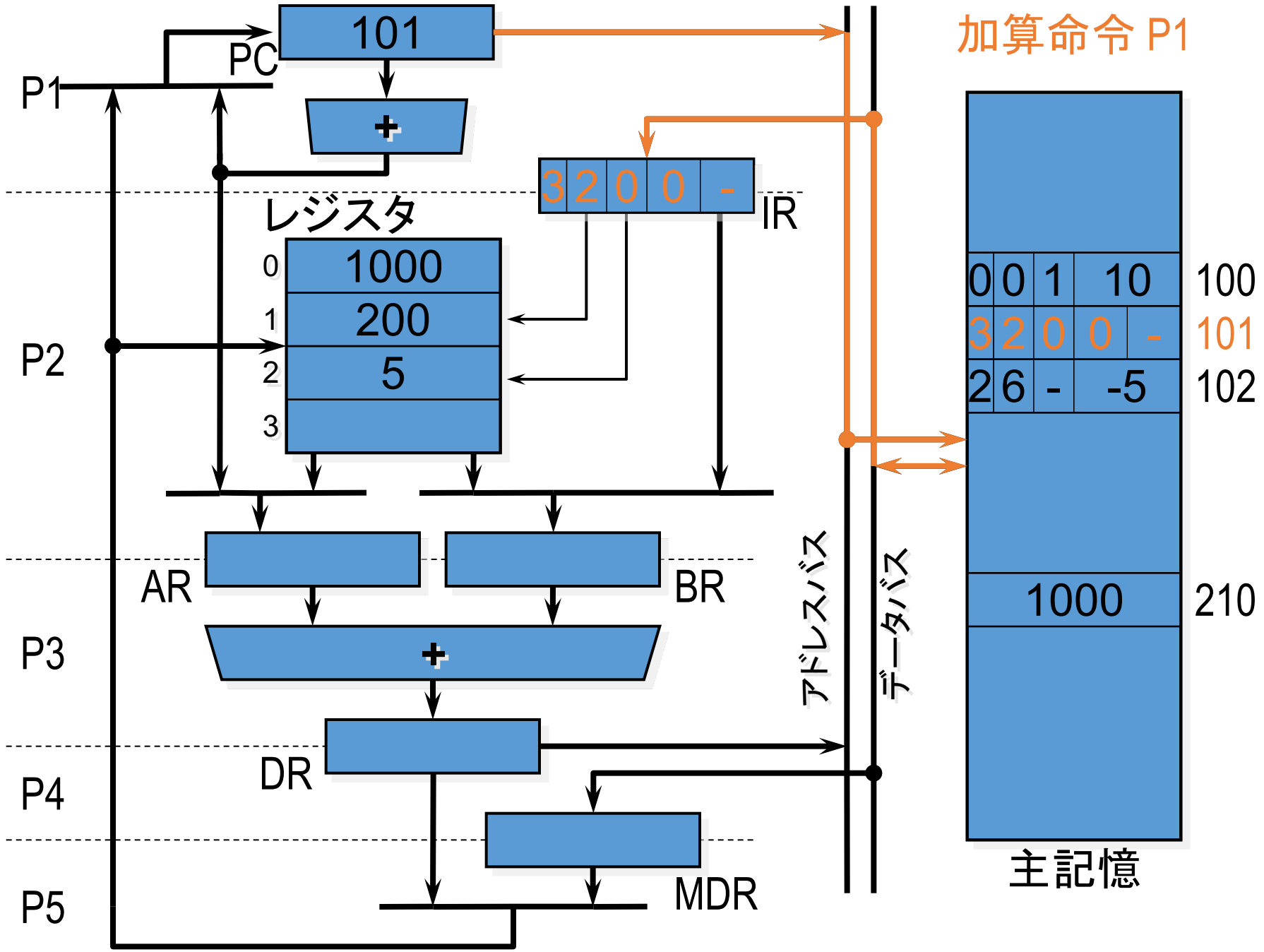

ロード命令
P1（命令フェッチ）
100
PC
P1
+
0 0 1 10
IR
レジスタ
0
1
200
2
5
3
P2
AR
BR
P3
+
DR
P4
MDR
P5
アドレスバス
データバス
0 0 1 10
100
3 2 0 0 -
101
2 6 - -5
102
1000
210
主記憶

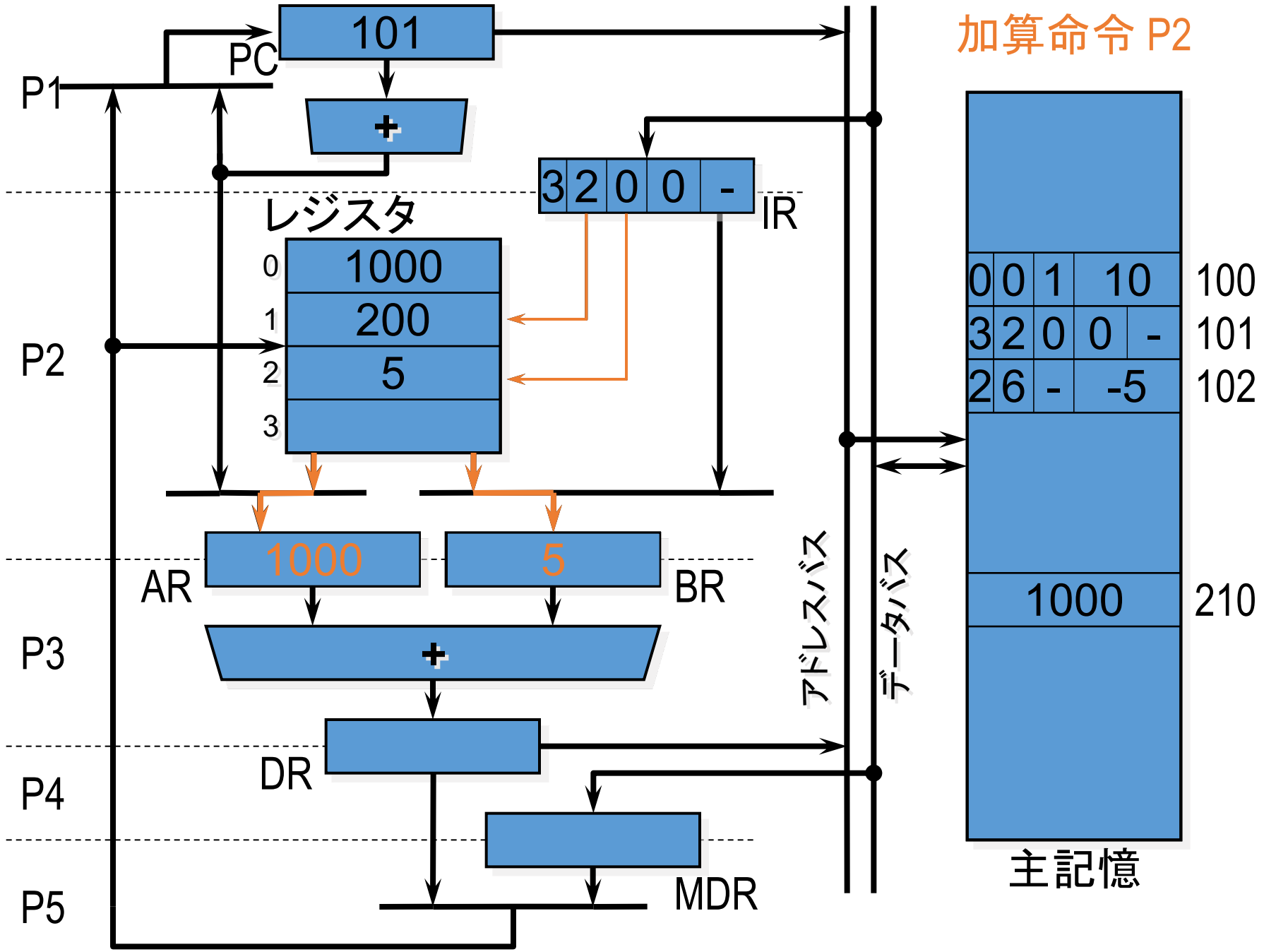

ロード命令
P2（レジスタ読出）
100
PC
P1
+
0 0 1 10
IR
レジスタ
0
1
200
2
5
3
P2
200
10
AR
BR
P3
+
DR
P4
MDR
P5
アドレスバス
データバス
0 0 1 10
100
3 2 0 0 -
101
2 6 - -5
102
1000
210
主記憶

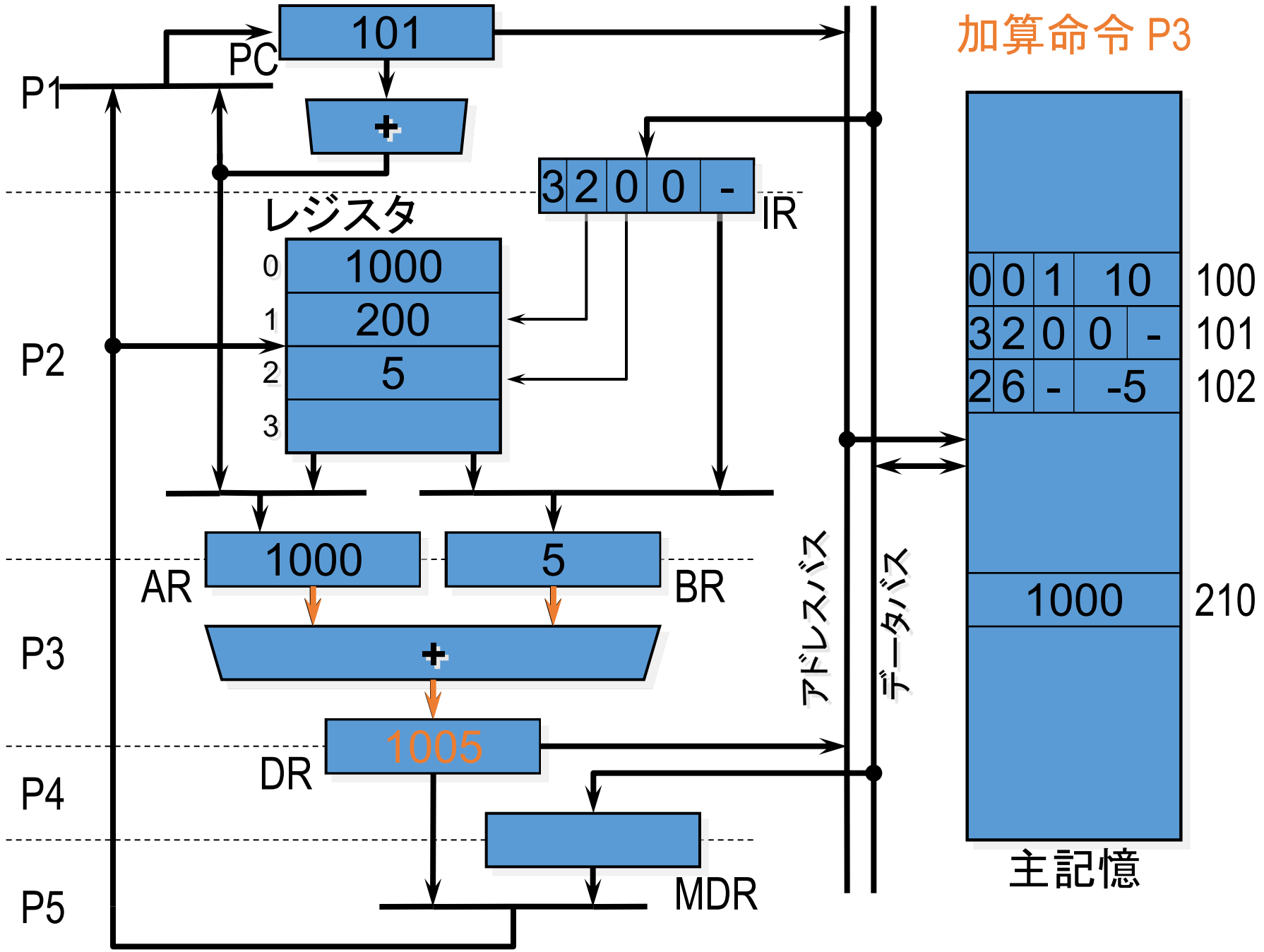

ロード命令
P3（演算）
100
PC
P1
+
0 0 1 10
IR
レジスタ
0
1
200
2
5
3
P2
200
10
AR
BR
P3
+
210
DR
P4
MDR
P5
アドレスバス
データバス
0 0 1 10
100
3 2 0 0 -
101
2 6 - -5
102
1000
210
主記憶

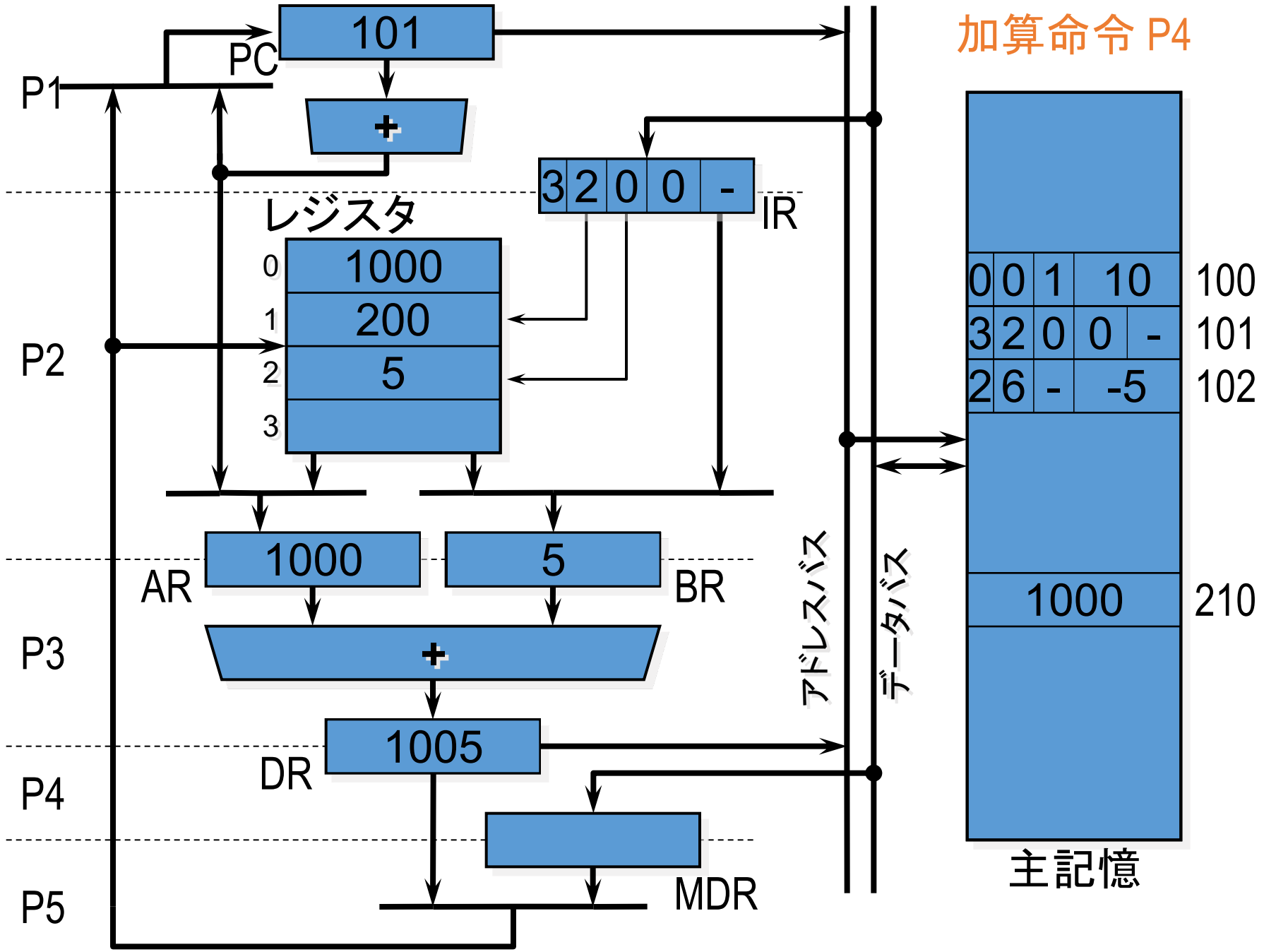

ロード命令
P4（主記憶アクセス）
100
PC
P1
+
0 0 1 10
IR
レジスタ
0
1
2
3
200
5
P2
200
10
AR
BR
P3
+
210
DR
P4
1000
MDR
P5
アドレスバス
データバス
0 0 1 10
100
3 2 0 0 -
101
2 6 - -5
102
1000
210
主記憶

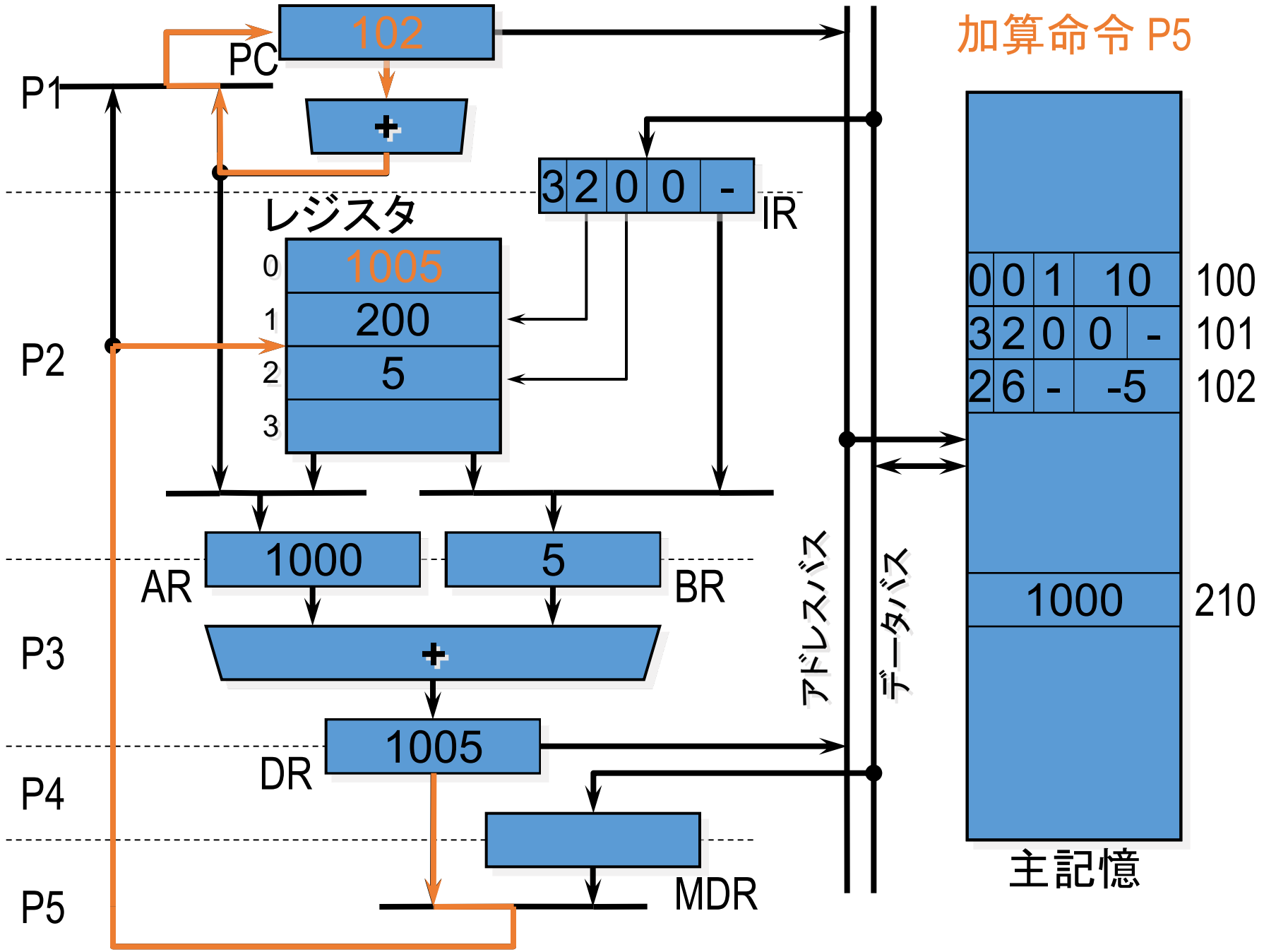

ロード命令
P5（レジスタ書き込み/
PC更新）
101
PC
P1
+
0 0 1 10
IR
レジスタ
0
1
2
3
1000
200
5
P2
200
10
AR
BR
P3
+
210
DR
P4
1000
MDR
P5
アドレスバス
データバス
0 0 1 10
3 2 0 0 -
2 6 - -5
1000
100
101
102
210
主記憶

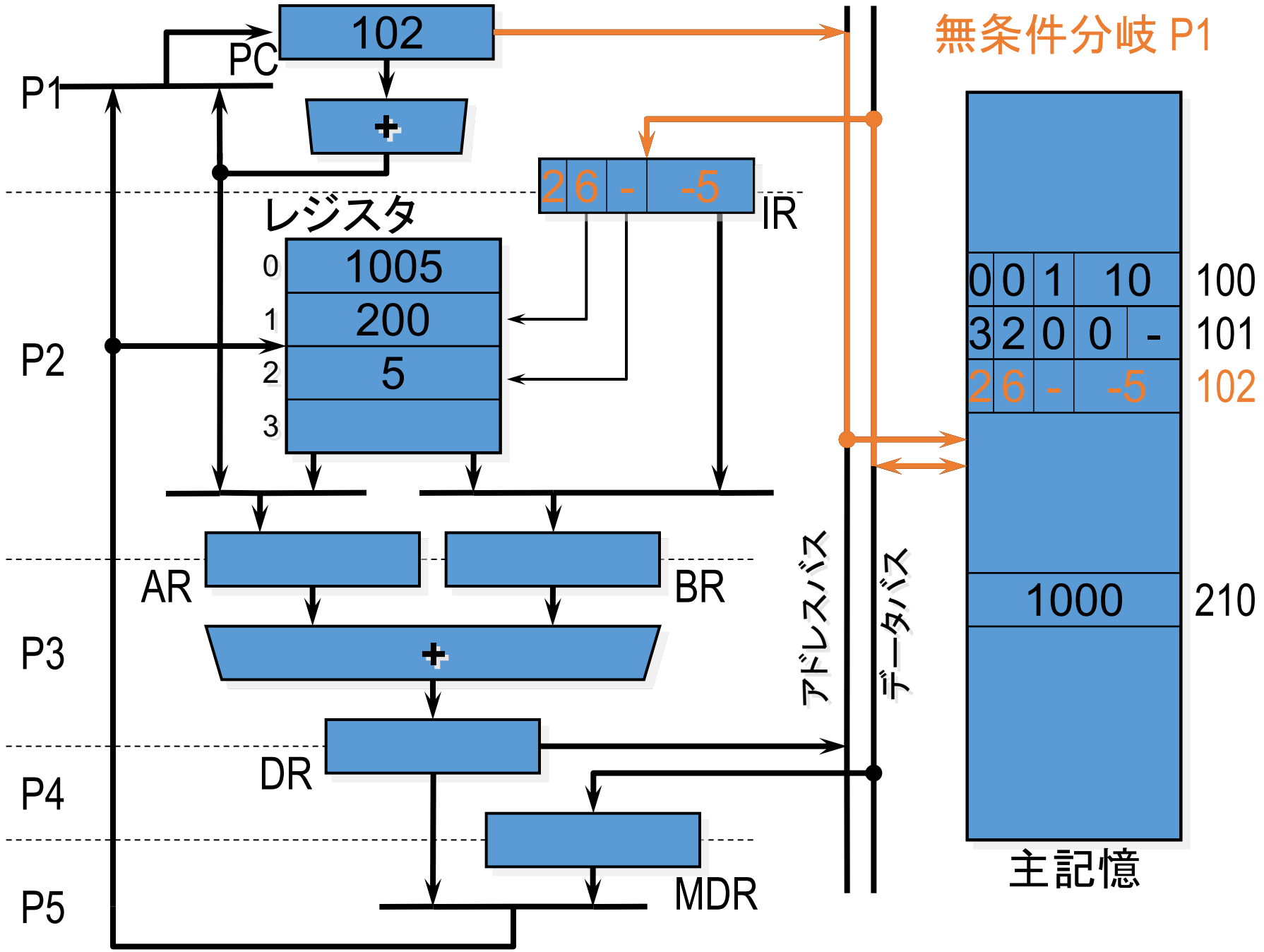

加算命令 P1
101
PC
P1
+
3200-
IR
レジスタ
0
1
2
3
1000
200
5
P2
AR
BR
P3
+
DR
P4
MDR
P5
アドレスバス
データバス
00110
320-
26-
-5
100
101
102
1000
210
主記憶

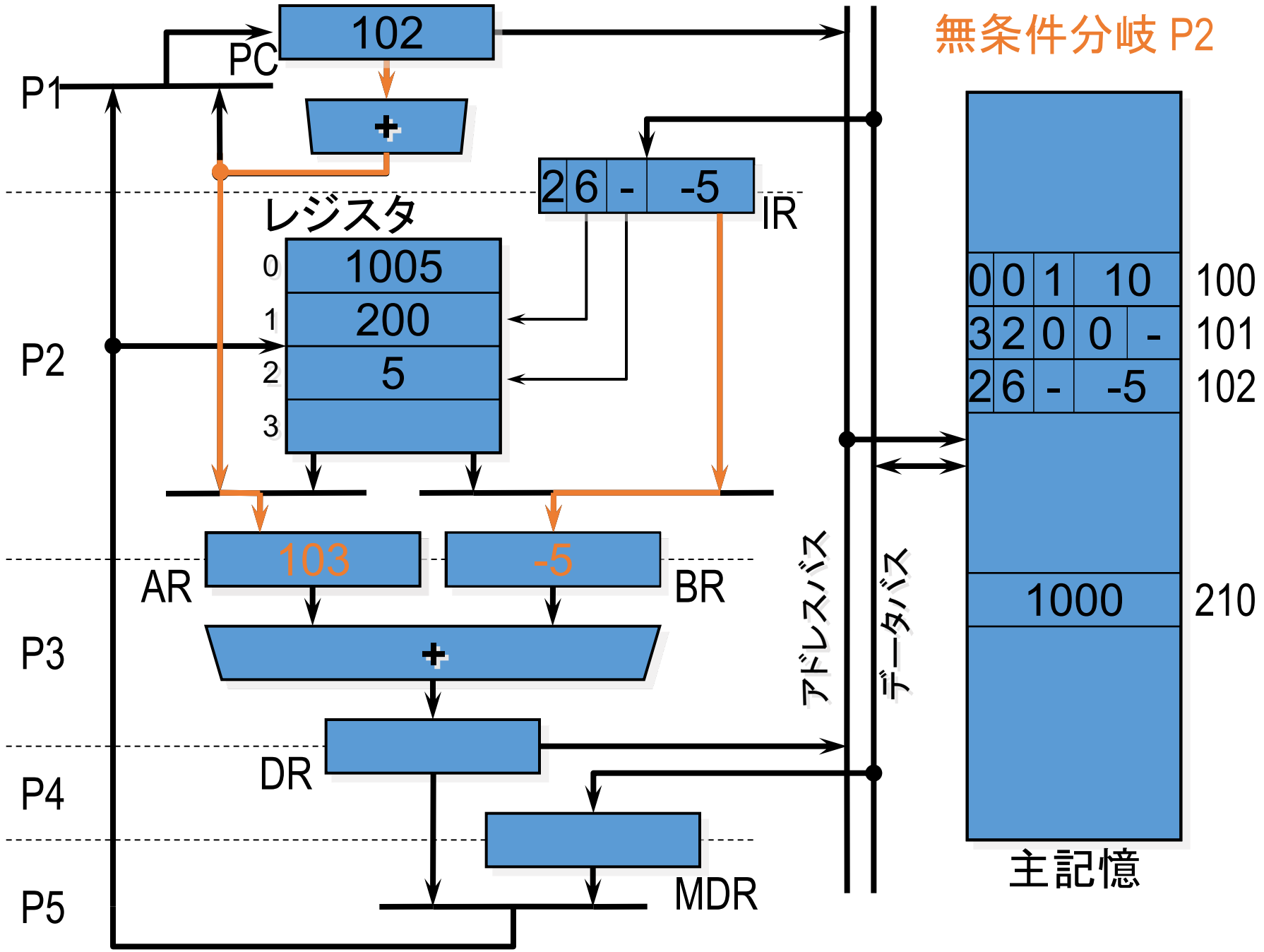

加算命令 P2
101
PC
P1
+
3200-
IR
レジスタ
0
1000
1
200
2
5
3
P2
1000
5
AR
BR
P3
+
DR
P4
MDR
P5
アドレスバス
データバス
00110 100
3200- 101
26-5 102
1000 210
主記憶

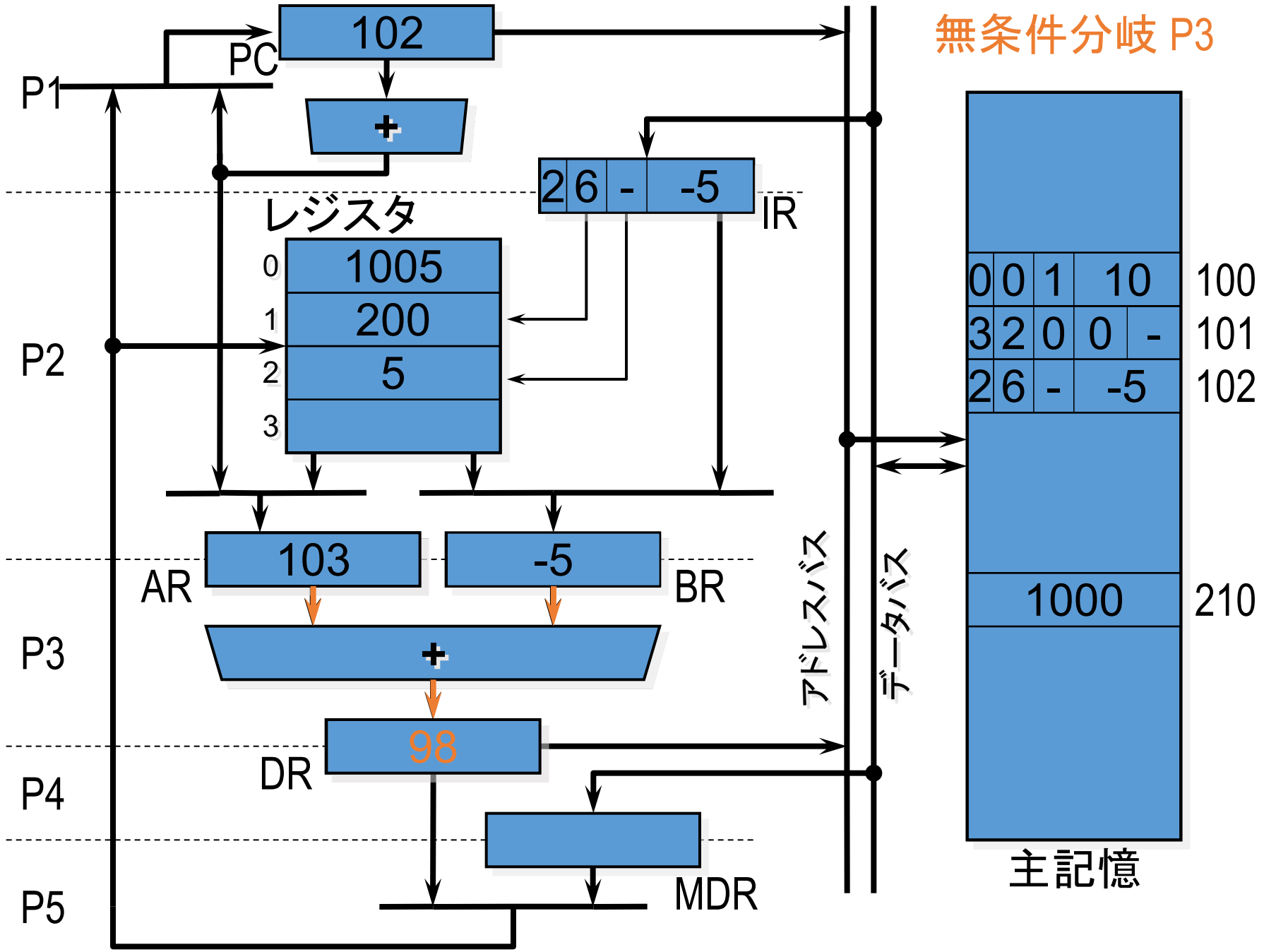

加算命令 P3
101
PC
P1
+
3200-
IR
レジスタ
0
1000
1
200
2
5
3
P2
1000
5
AR
BR
P3
+
1005
DR
P4
MDR
P5
アドレスバス
データバス
00110 100
3200- 101
26--5 102
1000 210
主記憶

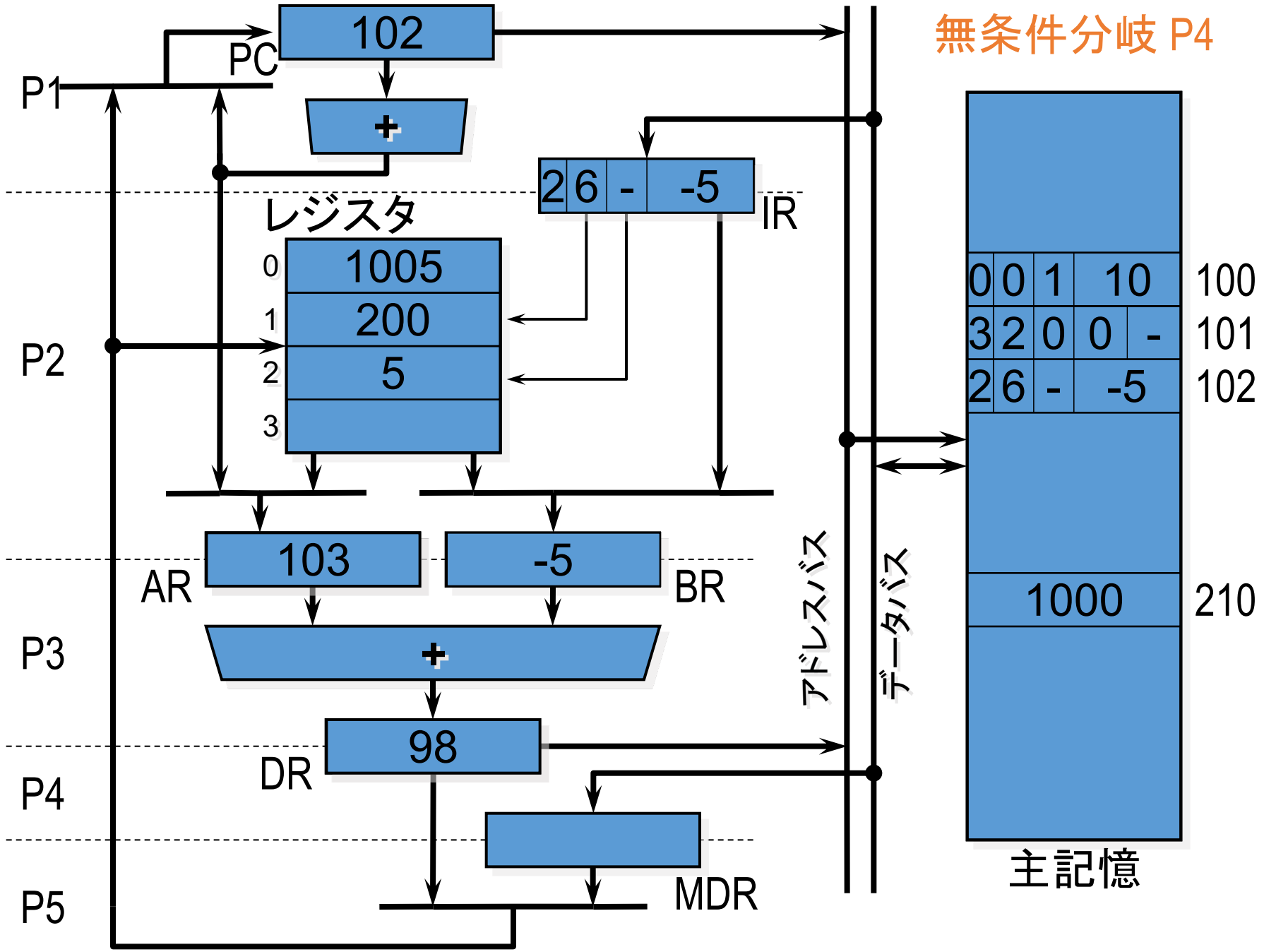

加算命令 P4
101
PC
P1
+
3200-
IR
レジスタ
0
1000
1
200
2
5
3
P2
1000
5
AR
BR
P3
+
1005
DR
P4
MDR
P5
アドレスバス
データバス
00110 100
3200- 101
26--5 102
1000 210
主記憶

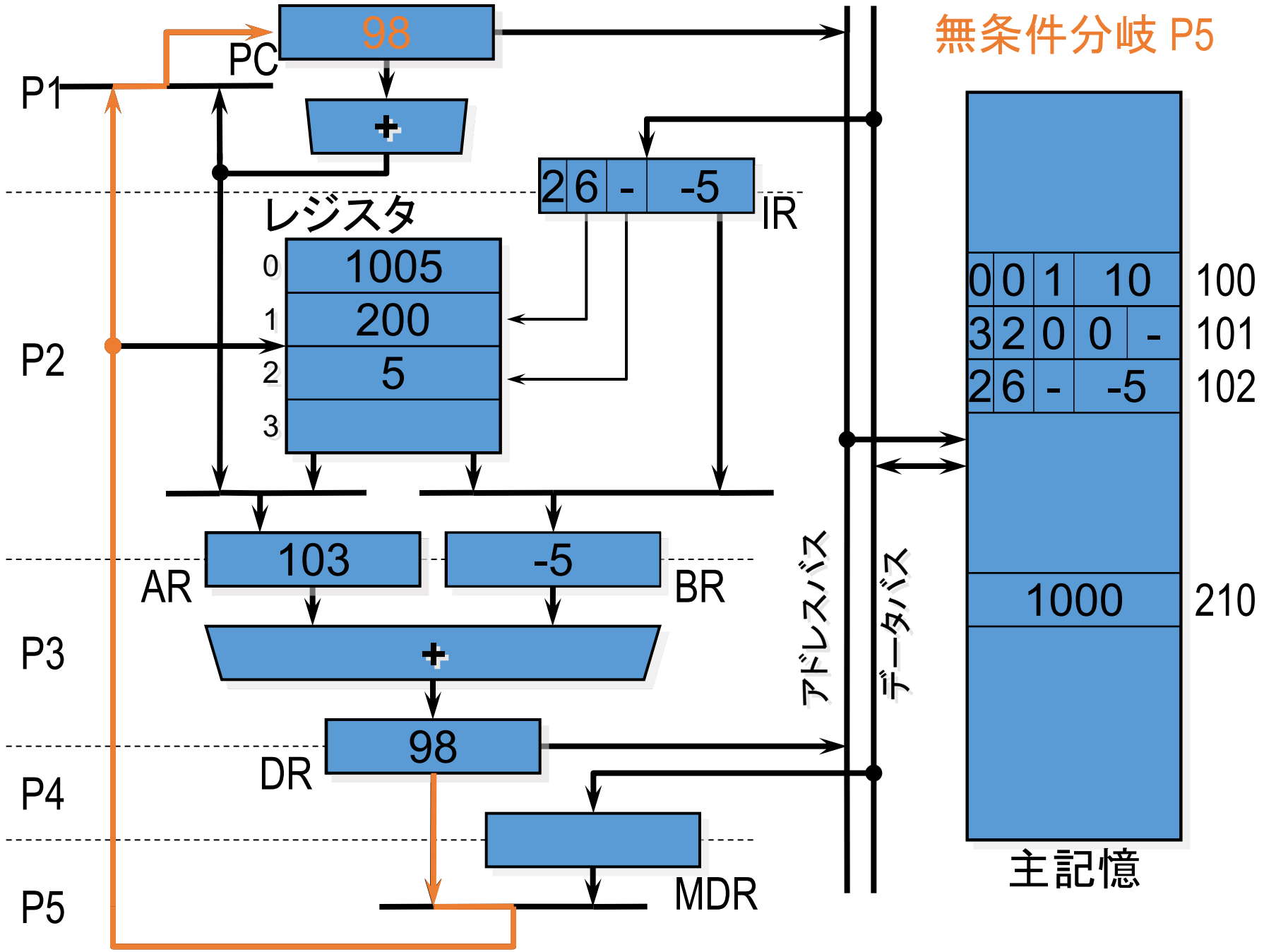

加算命令 P5
102
PC
P1
+
3200-
IR
レジスタ
0
1005
1
200
2
5
3
P2
1000
5
AR
BR
P3
+
1005
DR
P4
MDR
P5
アドレスバス
データバス
00110
100
3200-
101
26--5
102
1000
210
主記憶


無条件分岐 P1
102
PC
P1
+
26 - -5
IR
レジスタ
0
1005
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


無条件分岐 P2
102
PC
P1
+
26 - -5
IR
レジスタ
0
1005
1
200
2
5
3
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
+
26 - -5
IR
レジスタ
0
1005
1
200
2
5
3
P2
103
-5
AR
BR
P3
+
98
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 P4
102
PC
P1
+
26
-
-5
IR
レジスタ
0
1005
1
200
2
5
3
P2
103
AR
-5
BR
P3
+
98
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 P5
98
PC
P1
+
26 - -5
IR
レジスタ
0
1005
1
200
2
5
3
P2
103
AR
-5
BR
P3
+
98
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

課題とデモンストレーション

課題：SIMPLE/Bの機能拡張と性能評価

- 設計資料に記載のSIMPLE/Bから**何らかの拡張**を行って、拡張前と**比較評価**する
  - プログラムの実行がどれだけ高速化したか？
    - 最高クロック周波数、実行命令数、実行サイクル数
  - 追加で必要となったハードウェアは？
    - ゲート数（LUT数）
- 拡張の例　（SIMPLE設計資料の４章も参照）
  - 命令セットアーキテクチャの改良：
    命令の強化、新命令の追加、割り込みのサポート
  - マイクロアーキテクチャの改良：
    フェーズの並列実行(パイプライン化)、
    命令の並列実行（スーパースカラ）

デモンストレーションとレポート

- プロセッサ設計演習：　グループ単位(２～3人)
  - 中間デモ
    - 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
    - 最終的に作成を目指しているプロセッサの各種仕様書、考察等
  - 最終デモ
    - 設計し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
      - 応用プログラムの実行
  - 最終レポート
    - 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

- どう設計したか、何を使ったか、レポートに明記。
- 単に動くだけではなく、性能、回路規模、消費電力の最適化を目指す

コンテスト

- 「俺のプロセッサはすごいぜ！」ということを
  証明したい／歴史に名前を残したいあなたに…
- 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
- ぜひ参加して、これまでの記録を破ってください
  http://isle3hw.kuis.kyoto-u.ac.jp/contest/index.html

実験を進める上でのヒント

まず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

- トップダウン？ボトムアップ？
  - 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。
- プロトタイピング、マイルストーン、線表
  - 中間レポート時点「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
  - 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
  - テストベンチを作り込むことで、シミュレーションが自動化でき、手入力の作業を減らすことができる。
- 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。
  - ボードのスイッチやLEDを利用して内部信号をプローブできるようにする。
  - プロセッサ本体の外側にテスト用回路（プローブやスイッチ、表示系ドライバ）を構成。