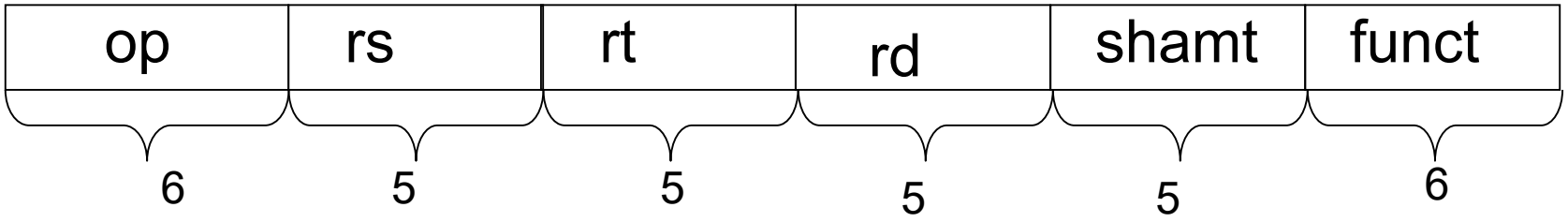
電子情報工学科実験III
ボードコンピュータ説明資料

孟林　小柳滋

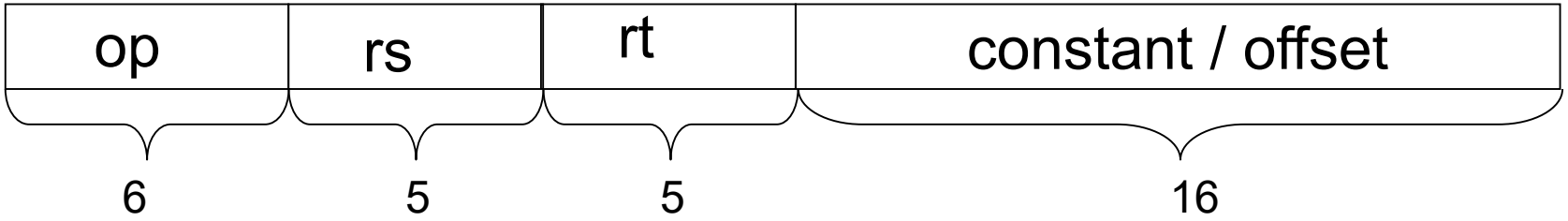
2020年度

MIPSの命令形式

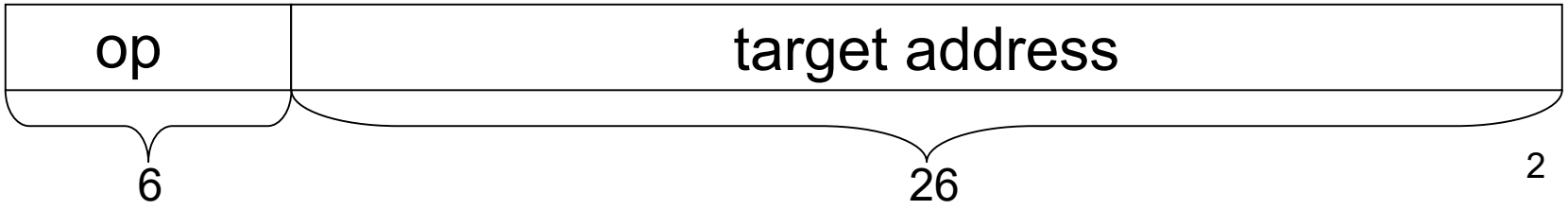
R形式（レジスタ演算）



I形式（定数演算、ロード・ストア、条件分岐）



J形式（無条件分岐）

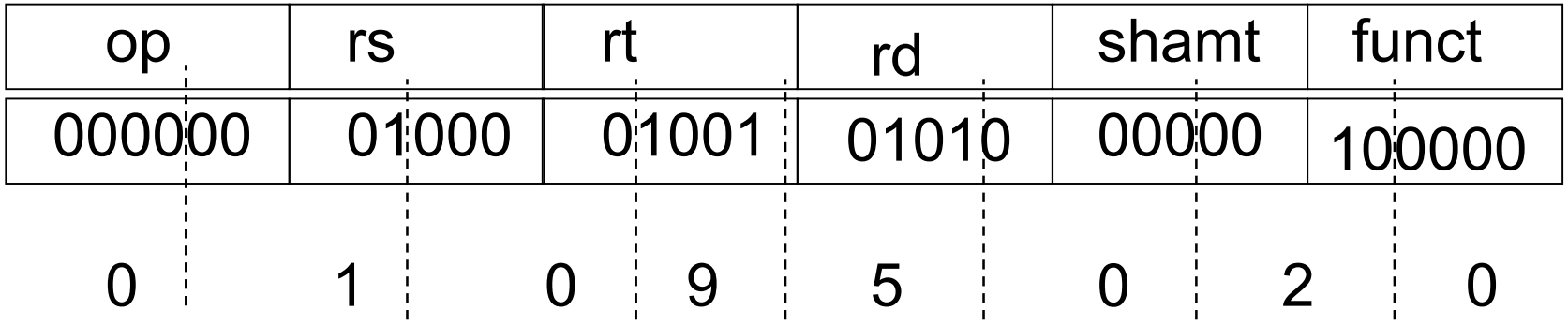


R形式の命令

ADD $10,$8,$9

rd rs rt

$10<=$8+$9
レジスタ8とレジスタ9の和を
レジスタ10に格納

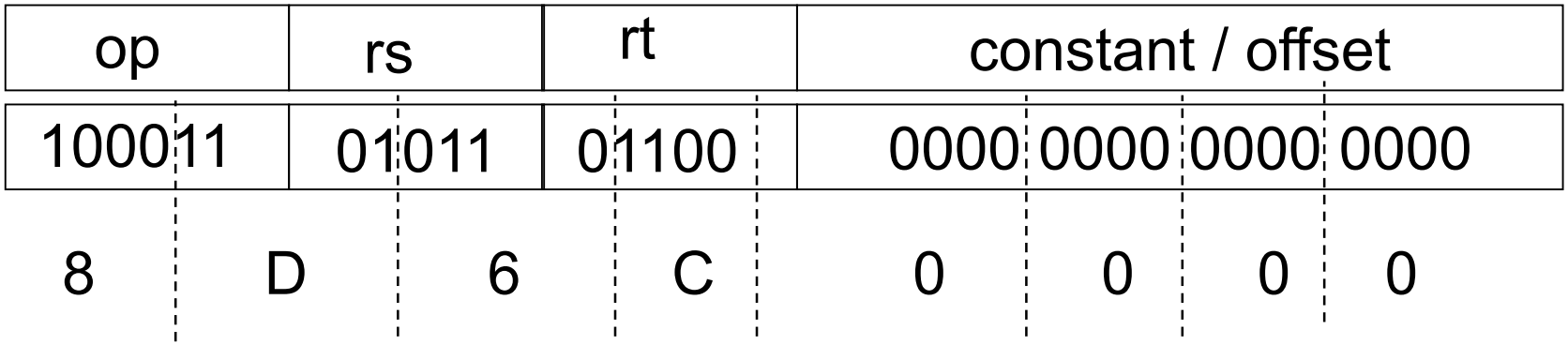


I形式の命令

LW $12, 0($11)

rt offset rs

$12<=M[$11+0]
レジスタ11とoffsetの和をアドレスとして
メモリを読み出し、レジスタ12に格納



条件分岐命令(BEQ、BNE)

BNE $8,$0,LOOP

rs　rt

if ($8!=$0) PC<=PC+4+offset*4
else　　PC<=PC+4

分岐先アドレス=PC+4+offset*4
offset=(分岐先アドレス−PC−4)/4

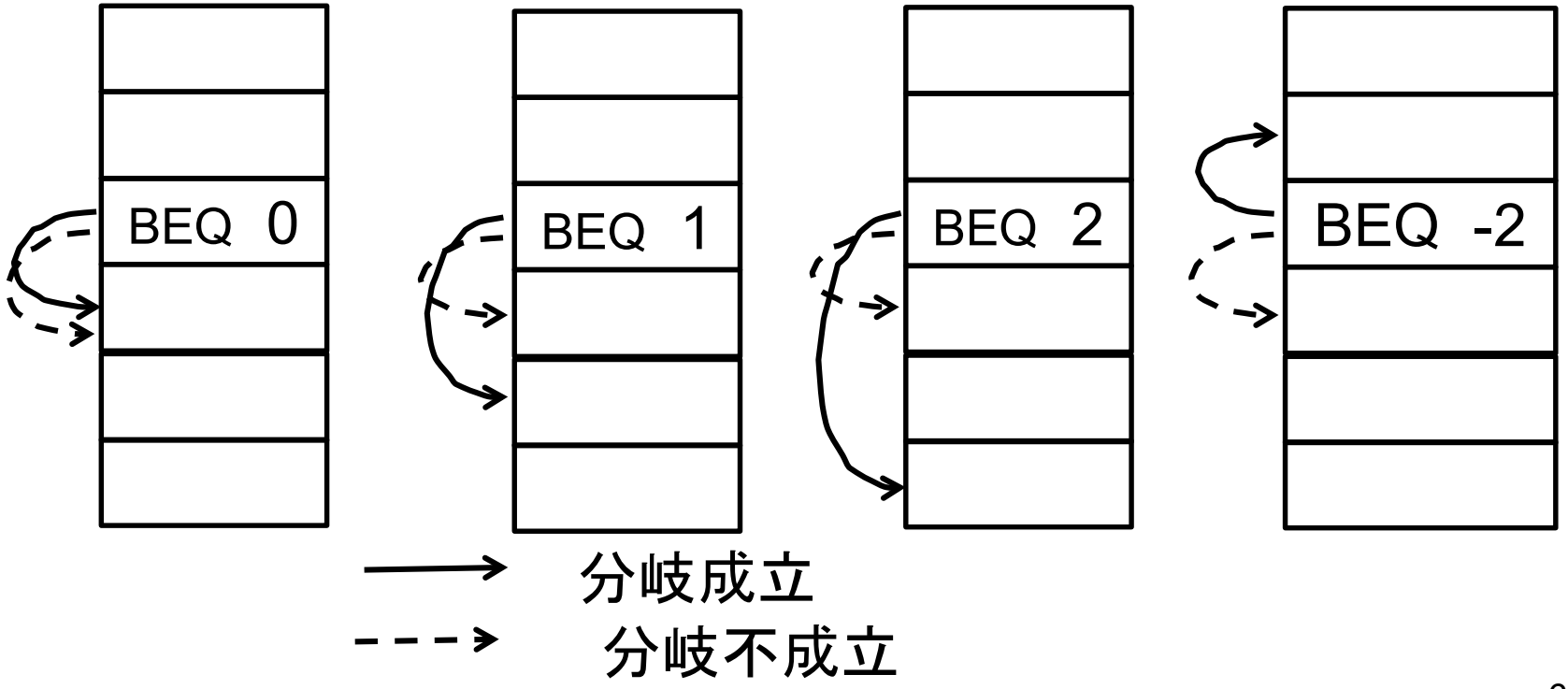
PC＝80000008、分岐先アドレス＝80000004のとき
offset=(80000004-80000008-4)/4=-2

op	rs	rt	constant / offset
000101	01000	00000	1111 1111 1111 1110
1	5	0	0	FFFE

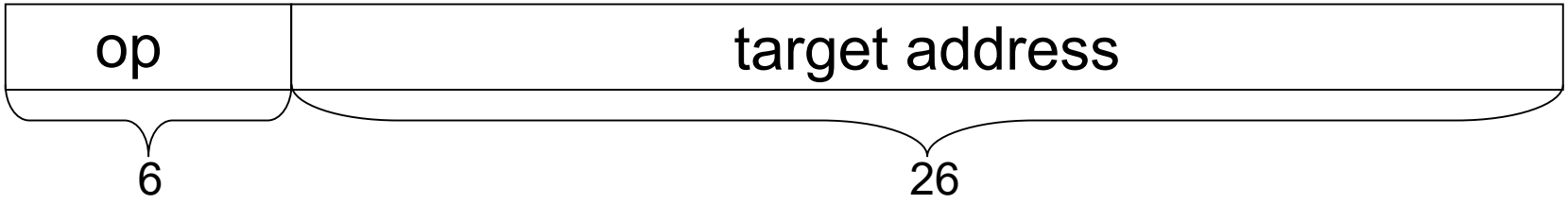
条件分岐命令でのアドレス指定

- 分岐条件成立のとき
  次のPC ← 現在のPC + 4 + (offset)*4
- 分岐条件不成立のとき
  次のPC ← 現在のPC + 4

PC相対アドレス



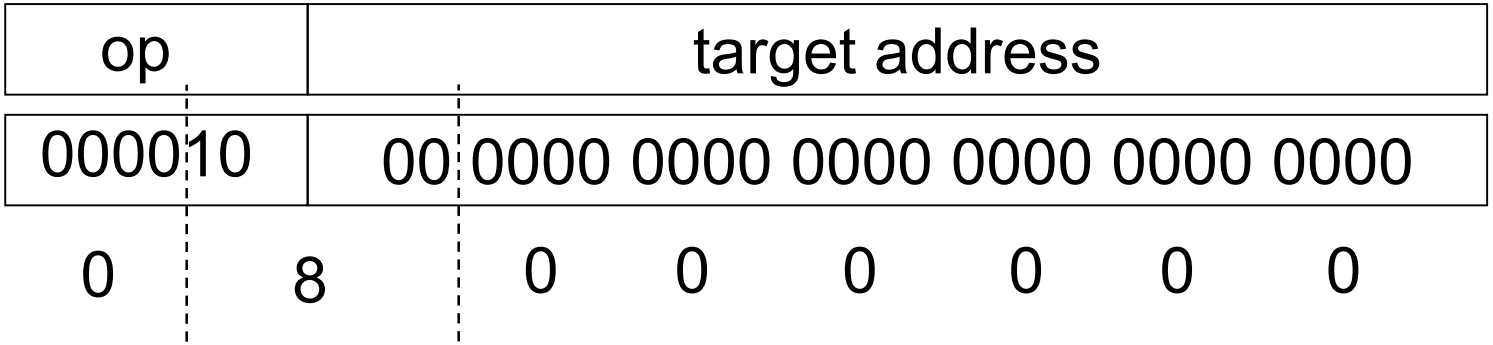
J形式の命令(J,JAL)



J　　START

分岐先アドレス=PCの上位4bitとtarget address*4を連結
target address=(分岐先アドレスの下位28bit)/4

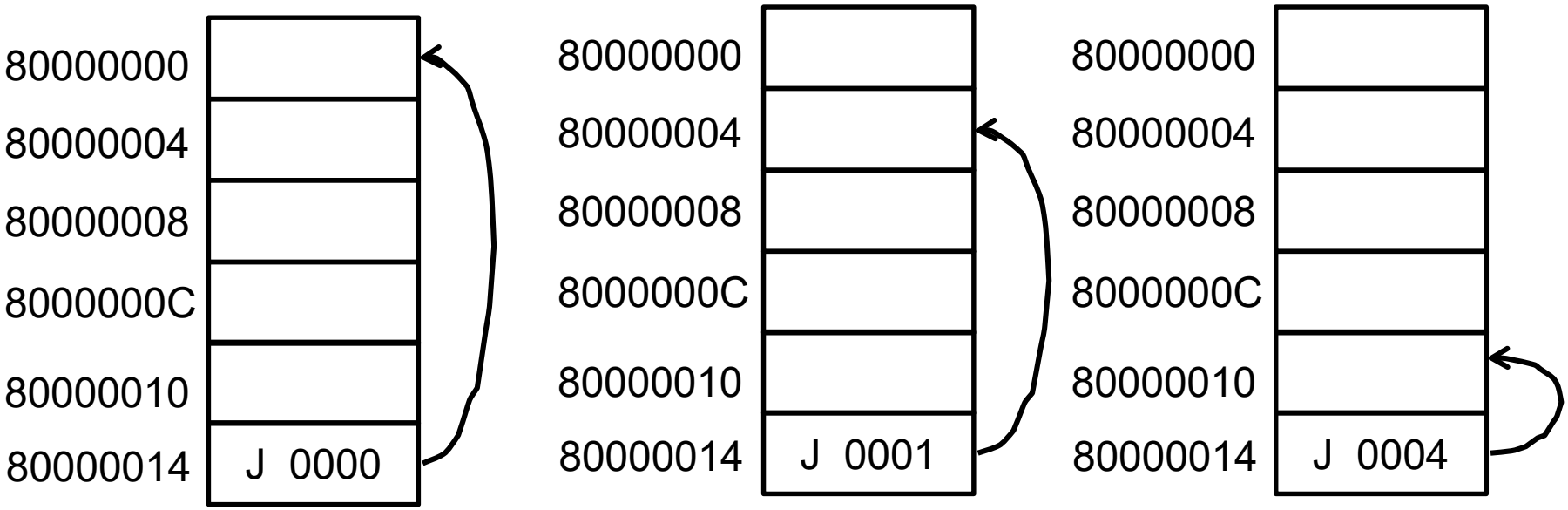
PC＝8000000C、分岐先アドレス＝80000000のとき
target address=0000000/4=0



無条件分岐命令でのアドレス指定

直接アドレス

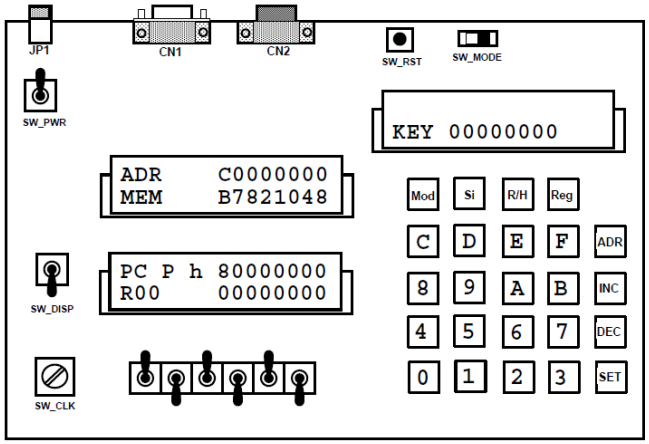
次のPC←{現在のPCの先頭4bit、26bitアドレス、00
}



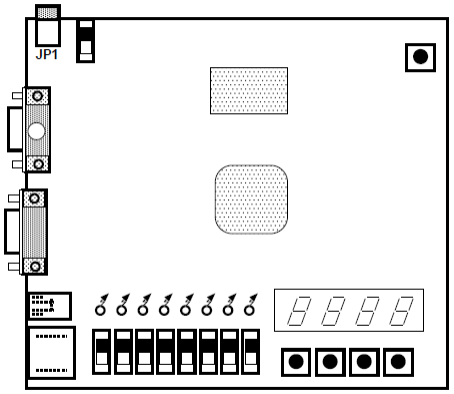
実験の進め方

- すべての課題について実験を行い、観測結果をテキストの表に記入する。
- 各自の進捗確認表に記載されている課題が終了したとき、TAに確認してもらう。
- すべての課題が終了したとき、小テストに取り組む。
- 研究課題については、レポートで取り組んでほしい。任意であるが、積極的な取り組みを評価する。
- 実験終了時は、RUECHIP1を片付け、進捗確認表をTAに提出する。

RUECHIP1



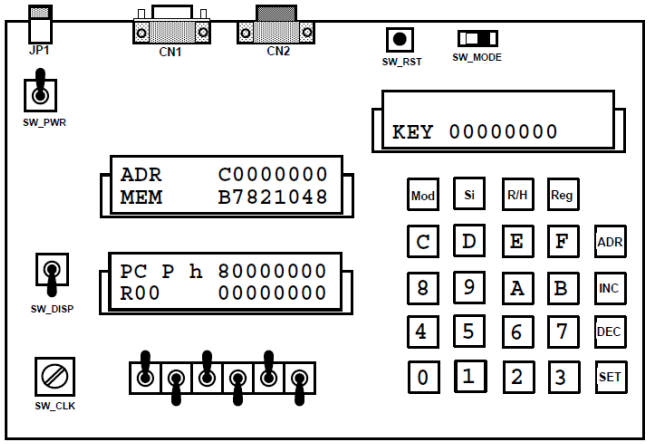
制御／観測ボード



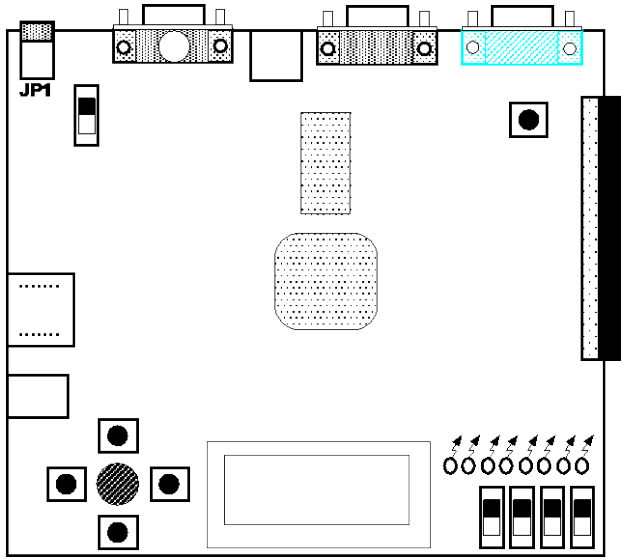
プロセッサボード

プロセッサボード上のFPGAにMIPSプロセッサが実現されている
制御／観測ボードより、プロセッサの内部が観測できる

RUECHIP1(Spatan-3E Starter Kit 版)



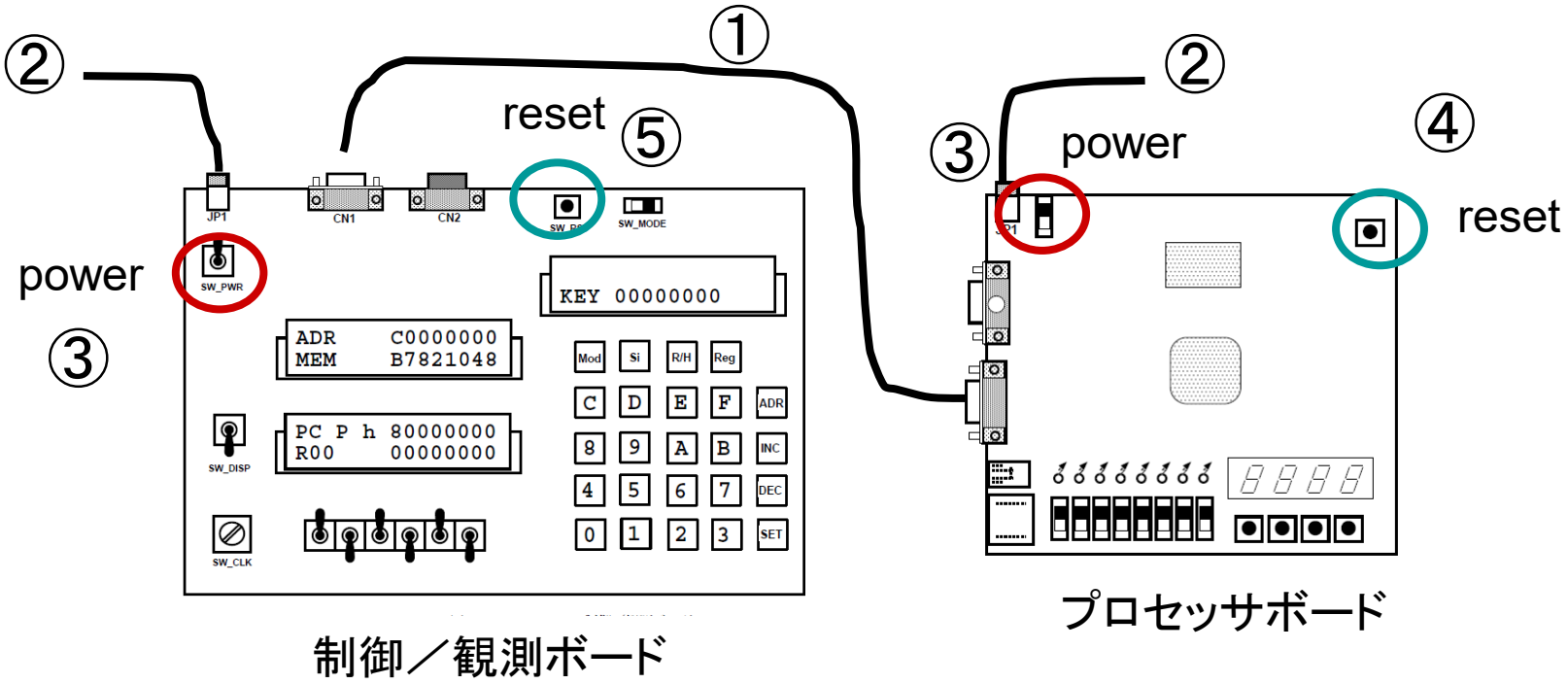
制御／観測ボード



プロセッサボード

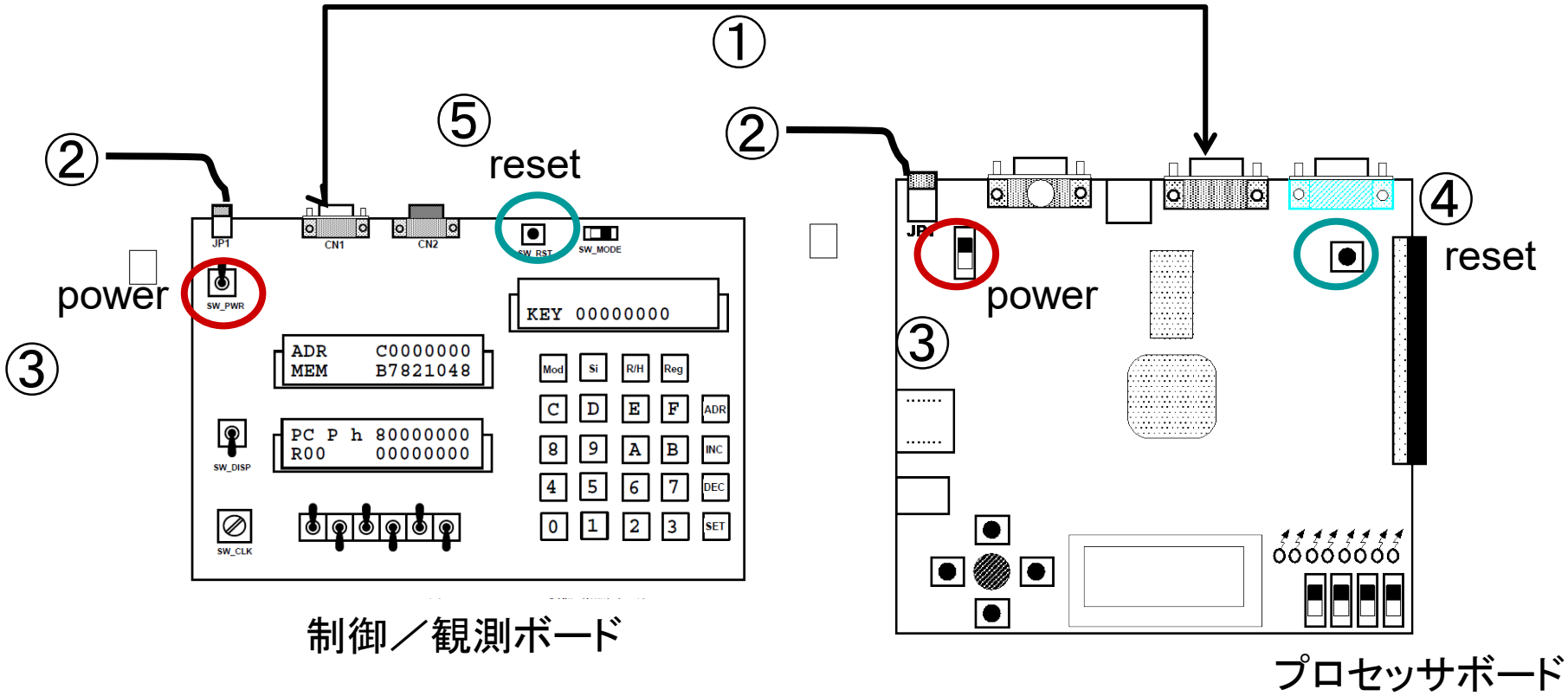
プロセッサボード上のFPGAにMIPSプロセッサが実現されている
制御／観測ボードより、プロセッサの内部が観測できる

RUECHIP1の接続



0. 2つのボードのpowerがoffであることを確認
1. プロセッサボードと制御/観測ボードをシリアルケーブルで接続
2. プロセッサボードと制御/観測ボードの電源ポートを接続
3. 2つのボードのpower on
4. プロセッサボードのreset
5. 制御／観測ボードのreset

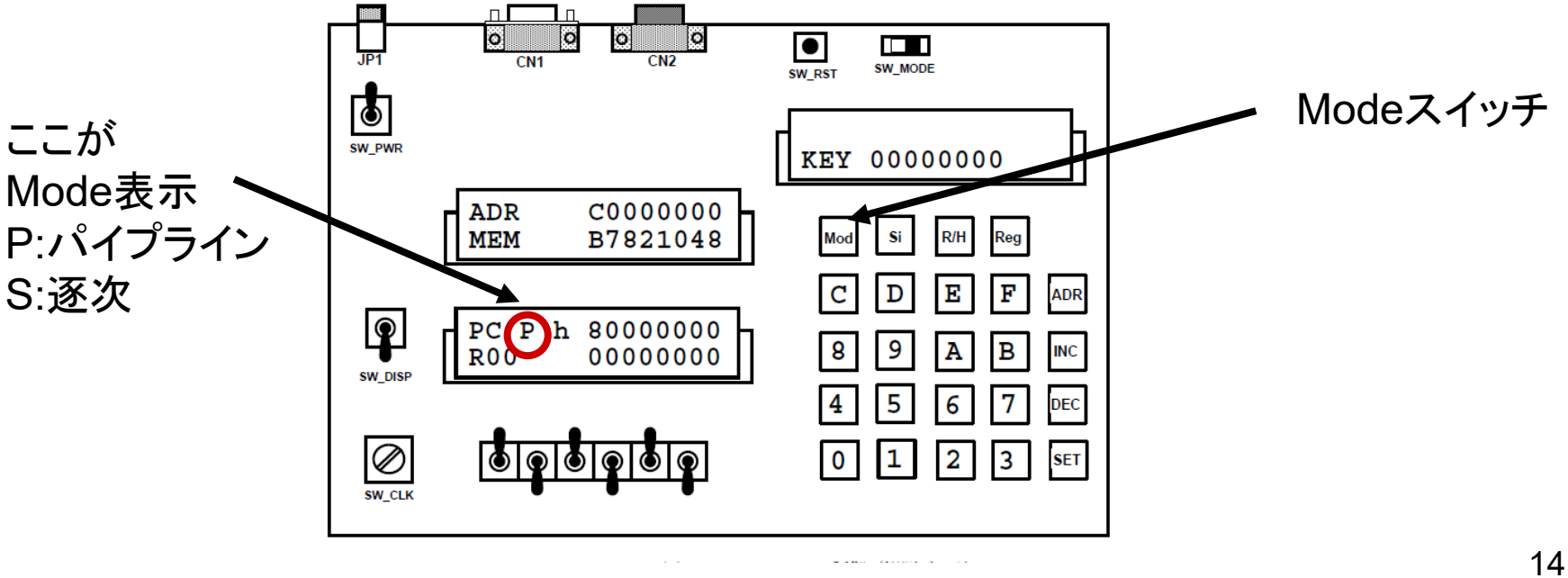
RUECHIP1(Spatan-3E Starter Kit 版)の接続



0. 2つのボードのpowerがoffであることを確認
1. プロセッサボードと制御/観測ボードをシリアルケーブルで接続
2. プロセッサボードと制御/観測ボードの電源ポートを接続
3. 2つのボードのpower on
4. プロセッサボードのreset
5. 制御／観測ボードのreset

逐次実行モードとパイプラインモード

- RUECHIP1では2つの実行モードを備えている。第1回は逐次実行モードに設定する。

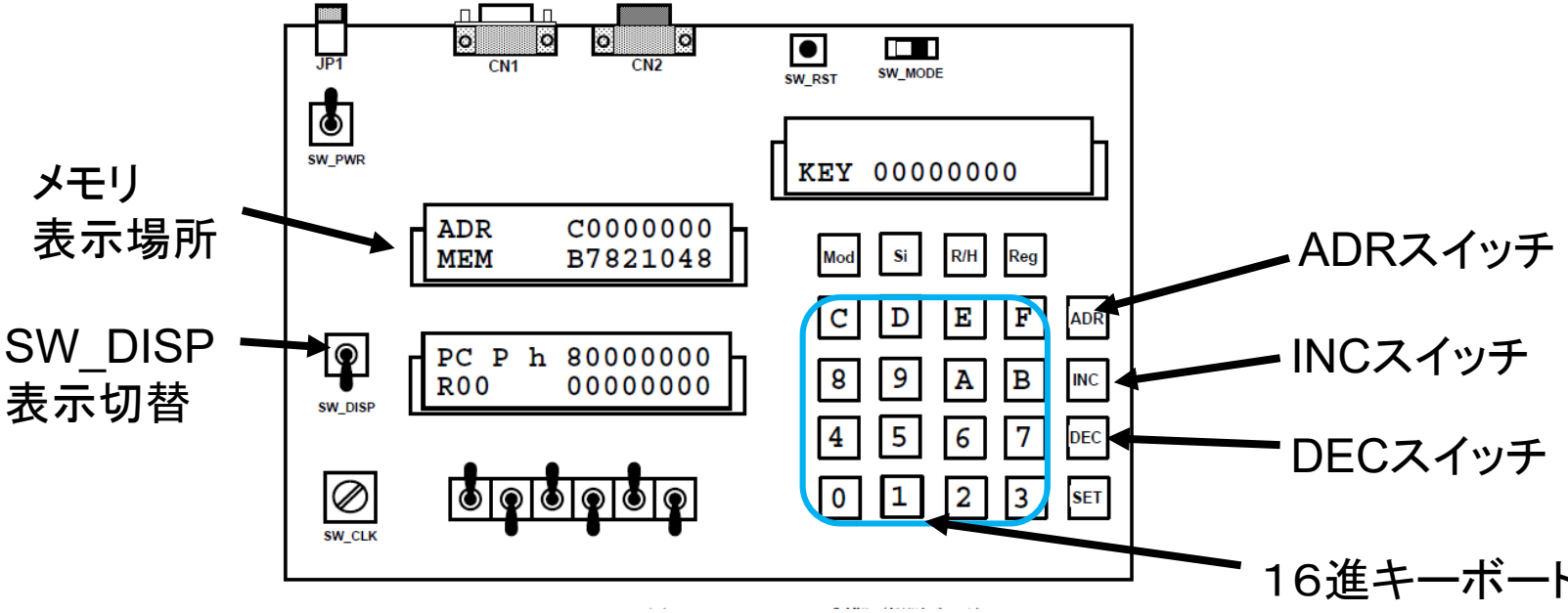


操作方法：メモリの読み出し

① 読み出したいメモリアドレスを16進キーボードより入力する。

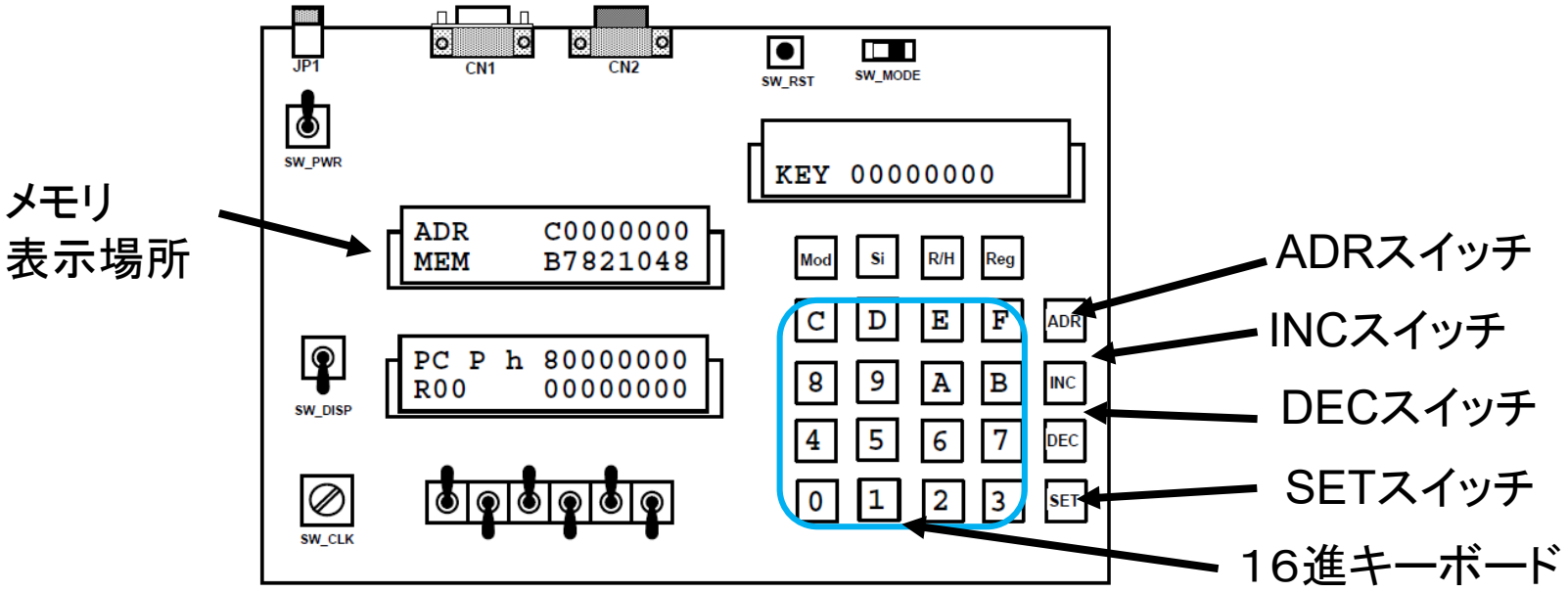
② ADRスイッチを押すとメモリのアドレスが設定され、そのアドレスと内容が表示される。（内容の表示はSW_DISPにより、16進数、命令形式を切り替えられる）

③ 連続番地を読み出すときはINCあるいはDECスイッチを押すと表示される。



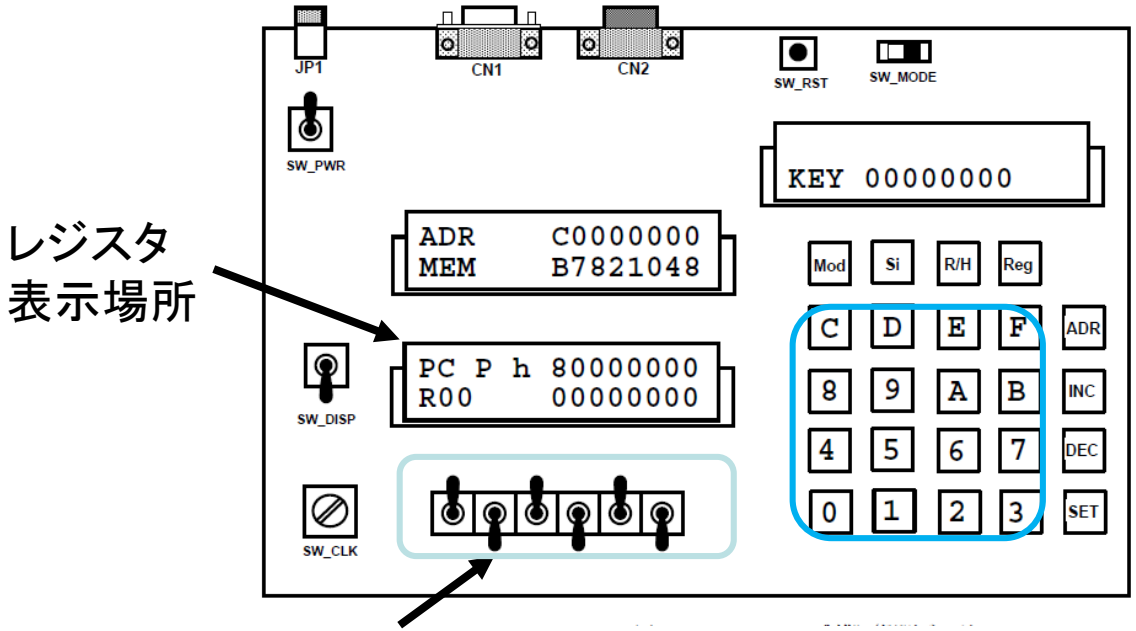
操作方法：メモリの書き込み

①書き込みたいメモリアドレスを16進キーボードより入力する。
②ADRスイッチを押すと、メモリのアドレスが設定される。
③書き込みたいデータを16進キーボードより入力する。（8桁）
④SETスイッチを押すと、指定されたメモリアドレスに書き込まれ、表示される。
⑤連続番地に書き込むときはINCあるいはDECスイッチを押し、③に戻る。



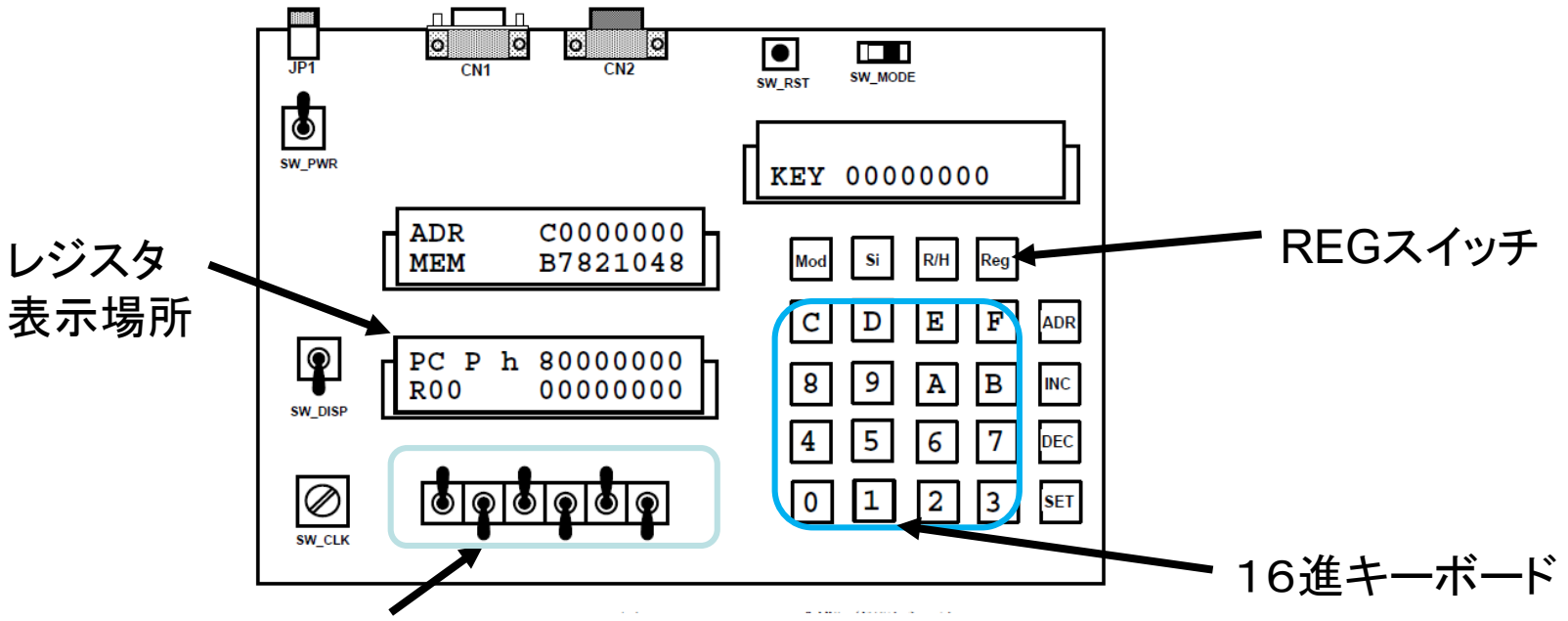
操作方法:レジスタの読み出し

①読み出したいレジスタアドレスをSW_SELに設定する。(6 bit)
レジスタアドレスはプログラムでは10進数、SW_SELでは2進数
0x00～0x1Fのとき汎用レジスタ、
0x20～0x3Fのとき内部レジスタ
② レジスタの内容が表示される。



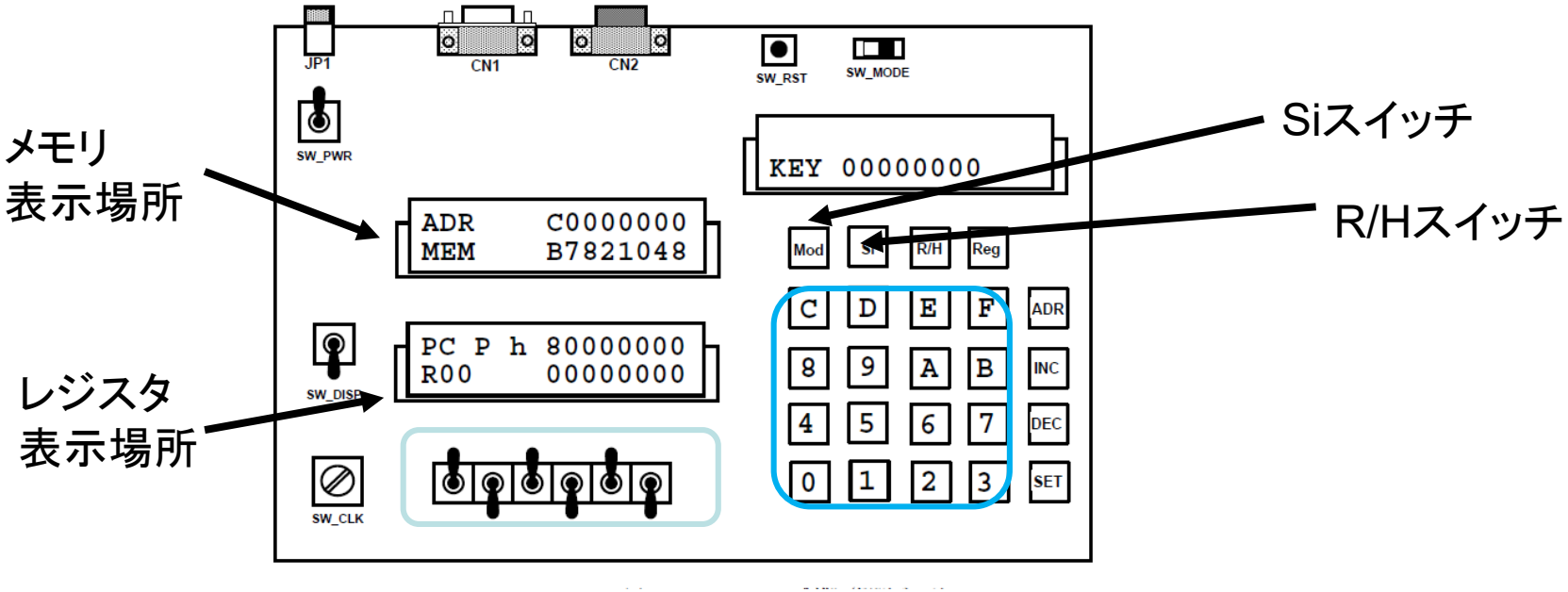
操作方法:レジスタの書き込み

①書き込みたいレジスタアドレスをSW_SELに設定する。(6 bit)
②書き込みたいデータを16進キーボードより入力する。(8桁)
③Regスイッチを押すとレジスタに書き込まれ、表示される。



操作方法:プログラムの実行

①実行したいプログラムをメモリに書き込む(80000000番地から)
②ボードをresetする　(PC=80000000に設定される)
③レジスタの初期設定をする
④R/Hスイッチを押す。(1クロック／1命令ずつ実行する場合はSiスイッチを押す)



第1週

課題1-1

- 例1-1を1命令ずつ実行し、観察結果の表を作成する

	アドレス	内容	アセンブリ言語	機能
①	8000 0000	2008 007A	ADDI $8, $0, 0x7A	$8 <= 0x7A
②	8000 0004	2009 0095	ADDI $9,$0,0x95	$9 <= 0x95
③	8000 0008	0109 5020	ADD $10,$8,$9	$10 <= $8 + $9
④	8000 000C	3c0b c000	LUI $11, 0xC000	$11 <= C000 0000
⑤	8000 0010	ad6a 0000	SW $10, 0($11)	M[$11] <= $10
⑥	8000 0014	8d6c 0000	LW $12, 0($11)	$12 <= M[$11]

例1-1プログラムの入力方法

- 16進キーボードで80000000と入力する
- ADRスイッチを押す（メモリアドレスの設定）
- 16進キーボードで2008007Aと入力する
- SETスイッチを押す（80000000番地に書き込み）
- INCスイッチを押す（メモリアドレスのインクリメント）
- 16進キーボードで20090095と入力する
- SETスイッチを押す（80000004番地に書き込み）

以降はこの繰り返し

全体の入力が終了した時点で、メモリの内容を読み出して確認すること

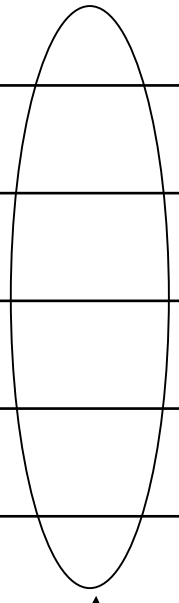
例1-1プログラムの実行方法

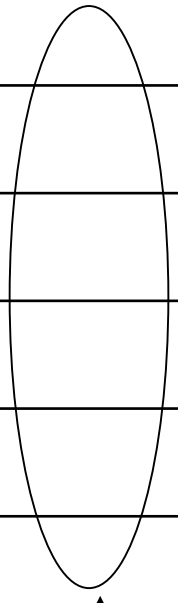
- RSTスイッチを押す（PCは80000000に設定される）
- 初期状態で調べたいレジスタアドレスをSW_SELに設定する

繰り返し
- Siスイッチを押す（1命令実行）
- 調べたいレジスタアドレスをSW_SELに設定し、内容を観測する

例1-1プログラムの観測方法

	初期値	命令①	命令②	命令③	命令④	命令⑤	命令⑥
PC							
$8							
$9							
$10							
$11							
$12							



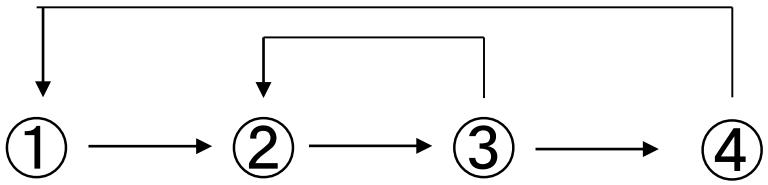


実行前のレジスタ値を観測する

1命令実行後のレジスタ値を観測する

例1-2：左シフトの繰り返し

	アドレス	内容	アセンブリ言語	機能
①	8000 0000	2008 0001	START: ADDI $8,$0,1	$8 <= 1
②	8000 0004	0008 4040	LOOP: SLL $8,$8,1	$8 <= $8 << 1
③	8000 0008	1500 FFFE	BNE $8,$0, LOOP	if $8 != 0 goto LOOP
④	8000 000C	0800 0000	J START	goto START



課題1-2

- 例1-2を1命令ずつ実行し、観察する
- 例1-2を連続実行し、レジスタ8の値が16進数で1 → 2 → 4 → 8 → 10 → 20 → 40 → 80 → ・・・ 10000000→20000000→

  40000000→80000000と変化し、1に戻ってこれを繰り返すことを確認せよ。
- R／Hスイッチを用いて連続実行する。実行状況が見えるように、クロック周波数を調整すること

課題1-2（続き）

- レジスタ8の値が16進数で80000000→40000000→20000000→・・・→10→8→4→2→1となり、80000000に戻って右シフトを繰り返すように例1-2を書き換え、実行せよ。

課題1-2：右シフトの繰り返し

	アドレス	内容	アセンブリ言語	機能
①	8000 0000		START:	$8 <= 8000 0000
②	8000 0004		LOOP: SRL $8,$8,1	$8 <= $8 >> 1
③	8000 0008	1500 FFFE	BNE $8,$0, LOOP	if $8 != 0 goto LOOP
④	8000 000C	0800 0000	J START	goto START

例1-3：総和を求める

	アドレス	内容	アセンブリ言語	機能
①	8000 0000	0000 4020	ADD $8,$0,$0	$8 <= 0
②	8000 0004	3C09 C000	LUI $9,0xC000	$9 <= C000 0000
③	8000 0008	8D2A 0000	LOOP: LW $10,0($9)	$10 <= M[$9]
④	8000 000C	1140 0003	BEQ $10,$0,HALT	if ($10==0) goto HALT
⑤	8000 0010	010A 4020	ADD $8,$8,$10	$8 <= $8 + $10
⑥	8000 0014	2129 0004	ADDI $9,$9,4	$9 <= $9 + 4
⑦	8000 0018	0800 0002	J LOOP	goto 8000 0008
⑧	8000 001C	0800 0007	HALT: J HALT	halt

$8：総和を保持
$9：メモリのポインタ
$10：メモリから読み出した値を保持

課題1-3

- メモリのC0000000番地から順に1,2,4,3,0を格納し、例1－3のプログラムを実行して、プログラムの動きを観測せよ。ループの繰り返しごとに、レジスタを観測して表を作成せよ。(PC=8000000Cのときに観測せよ)
- 実行前にメモリの初期設定が必要

  M[C0000000]<=1

  M[C0000004]<=2

  M[C0000008]<=4

  M[C000000C]<=3

  M[C0000010]<=0

観測表

レジスタ	1回目	2回目	3回目	4回目	5回目
$8					
$9					
$10					

課題1-4

- 例1-3の一部を修正し、最大値を求めるプログラムを作成せよ。

  M[C0000000]<=1

  M[C0000004]<=2

  M[C0000008]<=4

  M[C000000C]<=3

  M[C0000010]<=0

- PC=8000000Cのところで、レジスタ($8,$9,$10)の値を観測せよ。

課題1-4：最大値を求める

	アドレス	内容	アセンブリ言語	機能
①	8000 0000	0000 4020	ADD $8,$0,$0	$8 <= 0
②	8000 0004	3C09 C000	LUI $9,0xC000	$9 <= C000 0000
③	8000 0008	8D2A 0000	LOOP: LW $10,0($9)	$10 <= M[$9]
④	8000 000C	1140 000	BEQ $10,$0,HALT	if ($10==0) goto HALT
⑤	8000 0010			テスト
⑥	8000 0014			条件分岐
⑦	8000 0018	0140 4020	ADD $8,$0,$10	$8 <= $10
⑧	8000 001C	2129 0004	ADDI $9,$9,4	$9 <= $9 + 4
⑨	8000 0020	0800 0002	J LOOP	goto 8000 0008
⑩	8000 0024	0800 0009	HALT: J HALT	halt

分岐先が変わるのでoffsetに注意

乗除算

乗除算は倍長の計算が必要となり、特殊レジスタ{HI,LO}を用いる
HI：上位32bit、LO：下位32bit

乗算：{HI,LO}<= RS × RT
除算：LO<=RS ÷ RT,　剰余はHI

汎用レジスタと特殊レジスタ{HI,LO}間の転送命令を用いる

MFHI(move from HI)	rd<=HI
MFLO(move from LO)	rd<=LO
MTHI(move to HI)	HI<=rs
MTLO(move to LO)	LO<=rs

課題1-5：乗除算

レジスタ$8、$9、$11に適当な正の整数を設定し、$8と$9の積を$10に格納し、$10／ $11の商を$12、剰余を$13に格納するプログラムを作成せよ。ただし、すべて31ビット以内で表現できる整数とする。

$8 × $9　　=>　$10
$10 ÷ $11 =>　$12（商）、$13（剰余）

進捗確認

① 課題1-1についてTAによる確認
② 課題1-2についてTAによる確認
③ 課題1-3についてTAによる確認
④ 課題1-4についてTAによる確認

ノルマ

⑤ 課題1-5についてTAによる確認
⑥ 小テストの回答

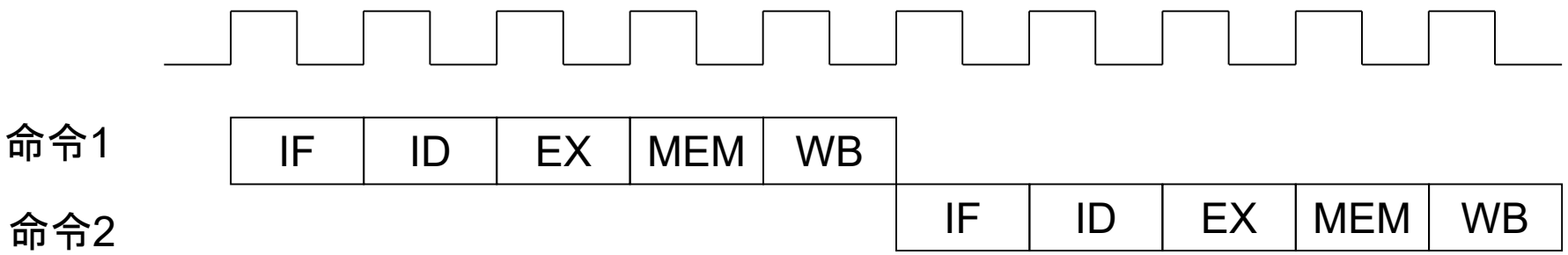
各課題において、動作内容をTAに説明すること

第2週

命令実行サイクル

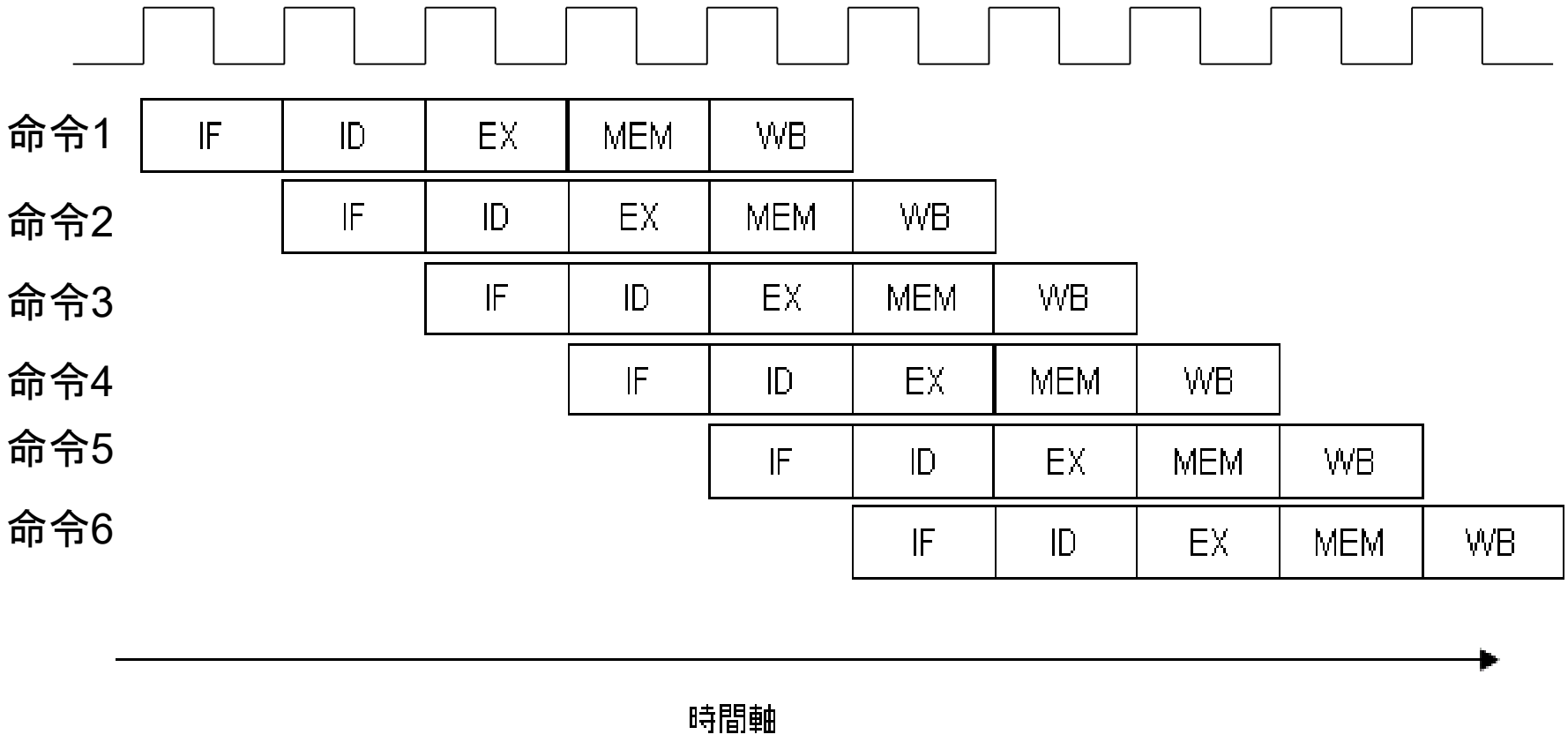
- IF（命令フェッチ）：プログラムカウンタ（PC）の示すアドレスより命令を読み出し、命令レジスタ（IR）に設定する。
- ID（命令デコード）：命令の解読を行い、ソースオペランドrsとrtを読み出す。
- EX（演算の実行）：ソースオペランドを用いて演算を行う。ロード/ストア命令ではアドレス計算を行う。
- MEM（メモリアクセス）：ロード命令ではデータメモリの読み出し、ストア命令ではデータメモリへの書き込みを行う。
- WB（レジスタの書込み）：演算結果をレジスタに書き込む。

非パイプライン処理



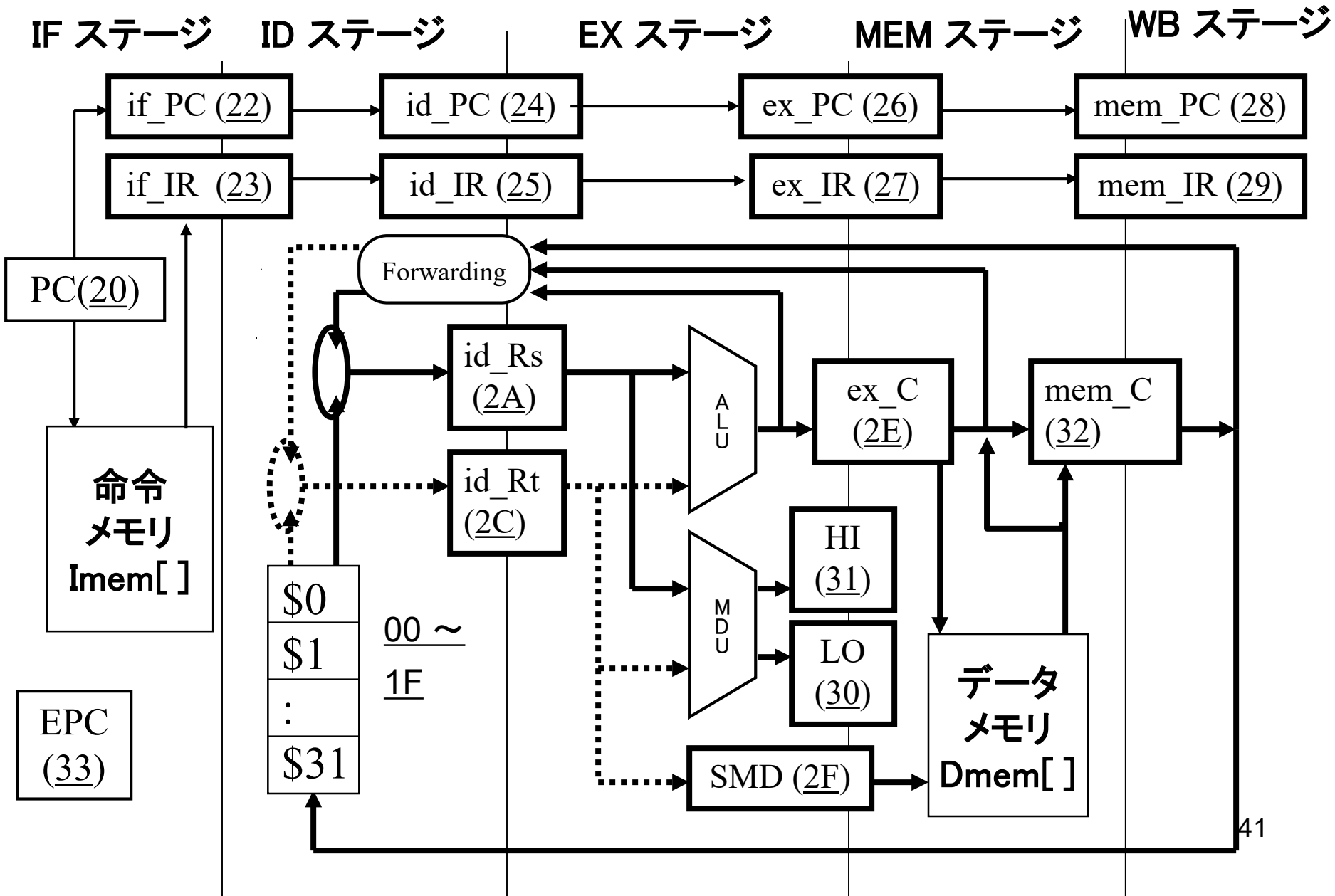
1クロックで1つのステージを実行する（マルチサイクルプロセッサ）
1命令の実行に5クロック要する

パイプライン処理



パイプライン化により、最大5倍性能が向上する

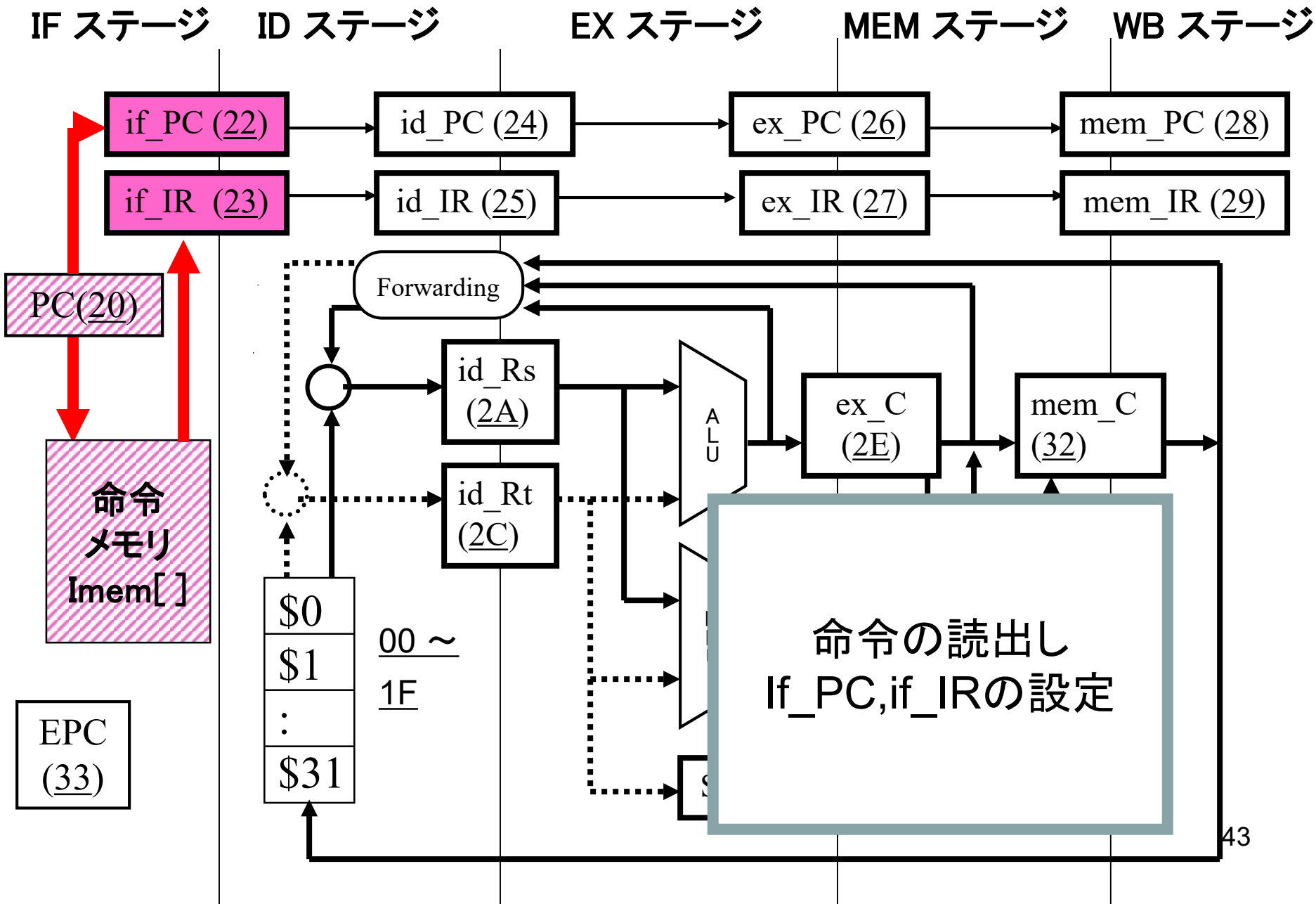
RUECHIP1のパイプライン構造



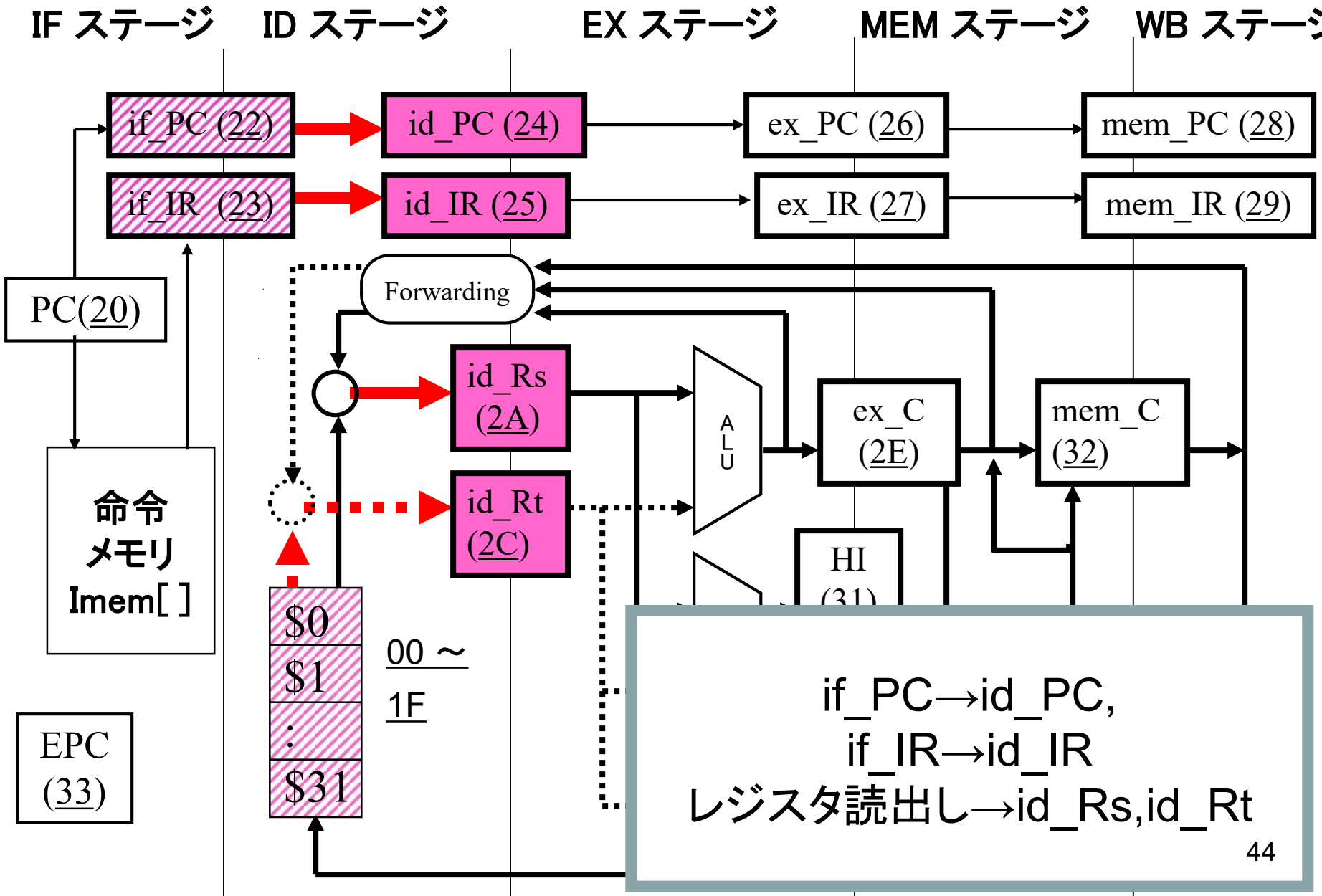
内部レジスタ(レジスタ番号0x20以降)

番号（16進数)	レジスタ名	役割
20	PC	プログラムカウンタ
22	if_PC	IFステージのプログラムカウンタ
23	if_IR	IFステージの命令レジスタ
24	id_PC	IDステージのプログラムカウンタ
25	id_IR	IDステージの命令レジスタ
26	ex_PC	EXステージのプログラムカウンタ
27	ex_IR	EXステージの命令レジスタ
28	mem_PC	MEMステージのプログラムカウンタ
29	mem_IR	MEMステージの命令レジスタ
2A	id_Rs	命令のrs部で指定されたレジスタの内容
2C	id_Rt	命令のrt部で指定されたレジスタの内容
2E	ex_C	ALUの演算結果を保持するレジスタ
2F	SMD	データメモリに書き込む内容を保持するレジスタ
30	LO	乗算用レジスタ
31	HI	乗算用レジスタ

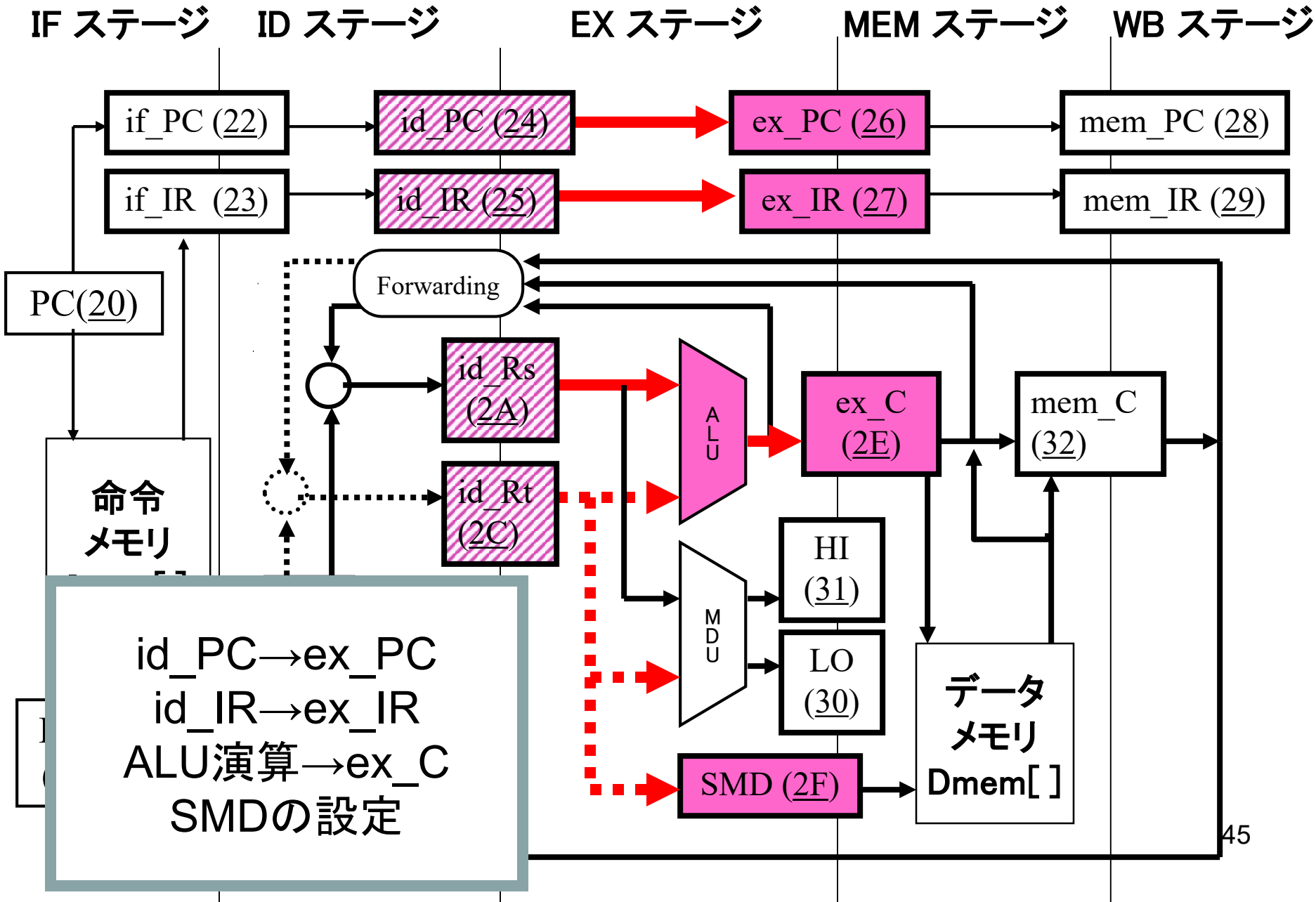
IFステージ


IF ステージ
ID ステージ
EX ステージ
MEM ステージ
WB ステージ
if_PC (22)
id_PC (24)
ex_PC (26)
mem_PC (28)
if_IR (23)
id_IR (25)
ex_IR (27)
mem_IR (29)
Forwarding
PC(20)
命令
メモリ
Imem[ ]
id_Rs
(2A)
id_Rt
(2C)
ALU
ex_C
(2E)
mem_C
(32)
$0
$1
:
$31
00 ～
1F
EPC
(33)
命令の読出し
If_PC,if_IRの設定

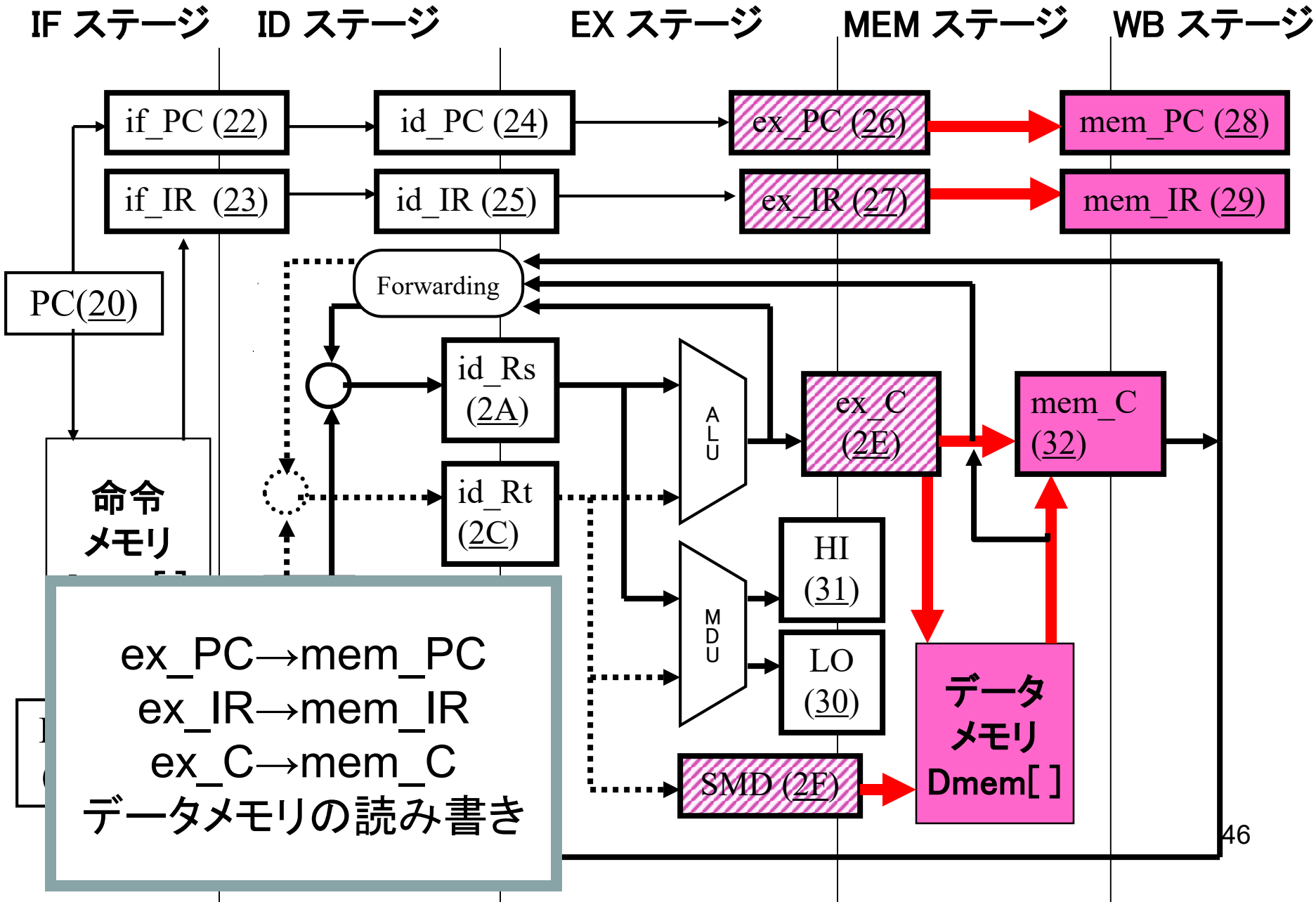
IDステージ


IF ステージ
ID ステージ
EX ステージ
MEM ステージ
WB ステージ
if_PC (22)
id_PC (24)
ex_PC (26)
mem_PC (28)
if_IR (23)
id_IR (25)
ex_IR (27)
mem_IR (29)
PC(20)
Forwarding
id_Rs (2A)
id_Rt (2C)
ALU
ex_C (2E)
mem_C (32)
HI (31)
命令メモリ Imem[ ]
$0
$1
:
$31
00 ～ 1F
EPC (33)
if_PC→id_PC,
if_IR→id_IR
レジスタ読出し→id_Rs,id_Rt

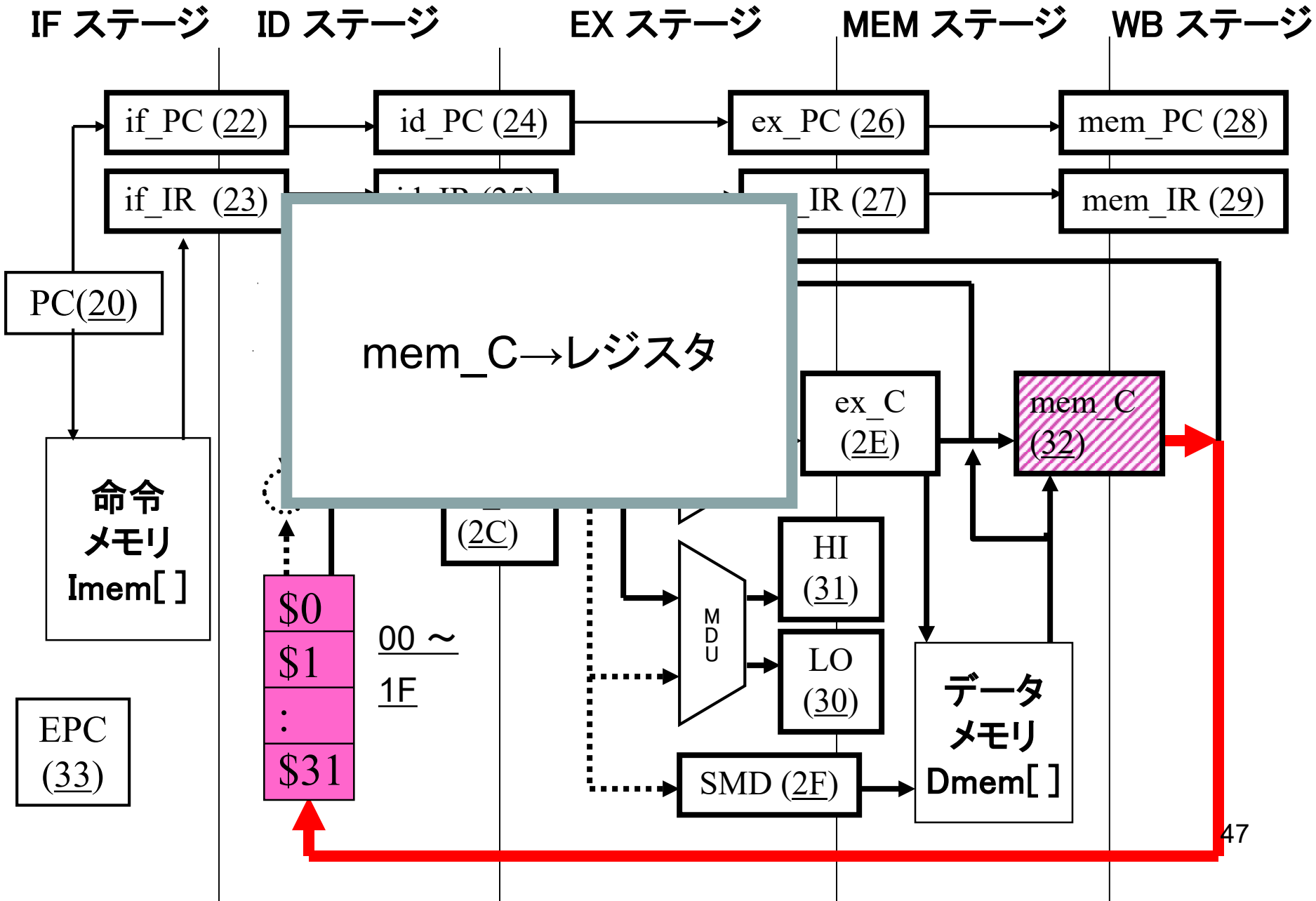
EXステージ


IF ステージ
ID ステージ
EX ステージ
MEM ステージ
WB ステージ
if_PC (22)
id_PC (24)
ex_PC (26)
mem_PC (28)
if_IR (23)
id_IR (25)
ex_IR (27)
mem_IR (29)
PC(20)
Forwarding
id_Rs (2A)
id_Rt (2C)
ALU
ex_C (2E)
mem_C (32)
命令メモリ
HI (31)
MDU
LO (30)
データメモリ Dmem[ ]
SMD (2F)
id_PC→ex_PC
id_IR→ex_IR
ALU演算→ex_C
SMDの設定

MEMステージ


IF ステージ
ID ステージ
EX ステージ
MEM ステージ
WB ステージ
if_PC (22)
id_PC (24)
ex_PC (26)
mem_PC (28)
if_IR (23)
id_IR (25)
ex_IR (27)
mem_IR (29)
PC(20)
Forwarding
id_Rs (2A)
id_Rt (2C)
ALU
ex_C (2E)
mem_C (32)
命令メモリ
HI (31)
MDU
LO (30)
SMD (2F)
データメモリ Dmem[ ]
ex_PC→mem_PC
ex_IR→mem_IR
ex_C→mem_C
データメモリの読み書き

WBステージ


IF ステージ
ID ステージ
EX ステージ
MEM ステージ
WB ステージ
if_PC (22)
id_PC (24)
ex_PC (26)
mem_PC (28)
if_IR (23)
IR (27)
mem_IR (29)
PC(20)
mem_C→レジスタ
ex_C
(2E)
mem_C
(32)
命令
メモリ
Imem[ ]
(2C)
HI
(31)
MDU
LO
(30)
$0
$1
:
$31
00 ～
1F
EPC
(33)
SMD (2F)
データ
メモリ
Dmem[ ]

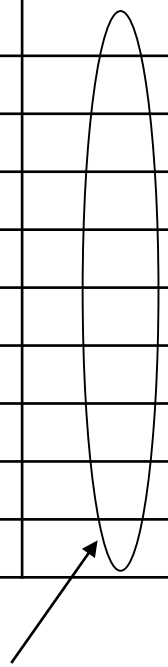
課題2-1：ADD命令の観測

	アドレス	内容	アセンブリ言語	機能
①	8000 0000	0109 5020	ADD $10, $8, $9	$10 <= $8 + $9
②	8000 0004	0000 0000	NOP	
③	8000 0008	0000 0000	NOP	
④	8000 000C	0000 0000	NOP	
⑤	8000 0010	0000 0000	NOP	
⑥	8000 0014	0000 0000	NOP	

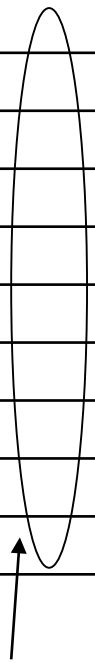
$8と$9に適当な値を設定する
初期状態、および1クロック入力毎にレジスタ等を観測する

観測表

レジスタ	初期値	1クロック	2クロック	3クロック	4クロック	5クロック
PC (20)						
IF_PC (22)						
ID_PC (24)						
EX_PC (26)						
MEM_PC (28)						
ID_RS (2A)						
ID_RT (2C)						
EX_C (2E)						
MEM_C (32)						
$10						



実行前のレジスタ値を観測



1クロック入力後のレジスタ値を観測

課題2-2：LW命令の観測

	アドレス	内容	アセンブリ言語	機能
①	8000 0000	8D09 0008	LW $9, 8($8)	$9 <= MM[$8+8]
②	8000 0004	0000 0000	NOP	
③	8000 0008	0000 0000	NOP	
④	8000 000C	0000 0000	NOP	
⑤	8000 0010	0000 0000	NOP	
⑥	8000 0014	0000 0000	NOP	

$8にC0000000を入力する
メモリのC0000008番地に適当な値を設定する
初期状態、および1クロック入力毎にレジスタ等を観測する

課題2-3：SW命令の観測

	アドレス	内容	アセンブリ言語	機能
①	8000 0000	AD09 0008	SW $9, 8($8)	MM[$8+8] <= $9
②	8000 0004	0000 0000	NOP	
③	8000 0008	0000 0000	NOP	
④	8000 000C	0000 0000	NOP	
⑤	8000 0010	0000 0000	NOP	
⑥	8000 0014	0000 0000	NOP	

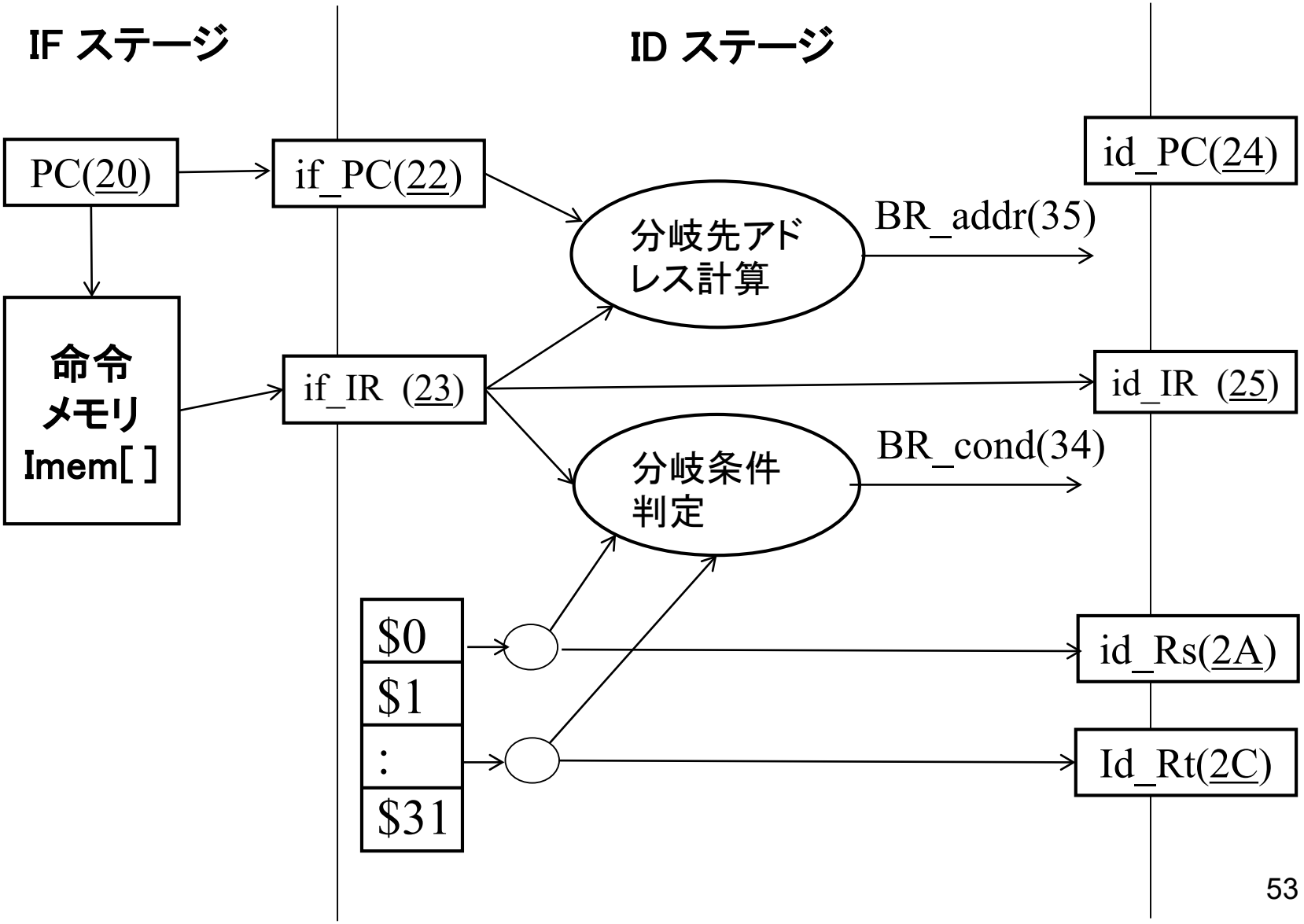
$8にC0000000を入力する
$9に適当な値を設定する
初期状態、および1クロック入力毎にレジスタ等を観測する

課題2-4：BEQ命令の観測

	アドレス	内容	アセンブリ言語	機能
①	8000 0000	1109 0002	BEQ $8, $9, L	If ($8==$9) goto L
②	8000 0004	0000 0000	NOP	
③	8000 0008	0000 0000	NOP	
④	8000 000C	0000 0000	NOP	
⑤	8000 0010	0000 0000	NOP	
⑥	8000 0014	0000 0000	NOP	

$8と$9に適当な値を設定する
分岐成立時と不成立時の2つの場合についてレジスタ等を観測する

分岐のメカニズム


IF ステージ
ID ステージ
PC(20)
if_PC(22)
id_PC(24)
分岐先アドレス計算
BR_addr(35)
命令メモリ Imem[ ]
if_IR (23)
id_IR (25)
分岐条件判定
BR_cond(34)
$0
$1
:
$31
id_Rs(2A)
Id_Rt(2C)

課題2-5：複数命令の観測

	アドレス	内容	アセンブリ言語	機能
①	8000 0000	0109 5020	ADD $10, $8, $9	$10 <= $8 + $9
②	8000 0004	016C 6820	ADD $13, $11, $12	$13 <= $11 + $12
③	8000 0008	01CF 8020	ADD $16, $14, $15	$16 <= $14 + $15
④	8000 000C	0000 0000	NOP	
⑤	8000 0010	0000 0000	NOP	
⑥	8000 0014	0000 0000	NOP	
⑦	8000 0018	0000 0000	NOP	
⑧	8000 001C	0000 0000	NOP	

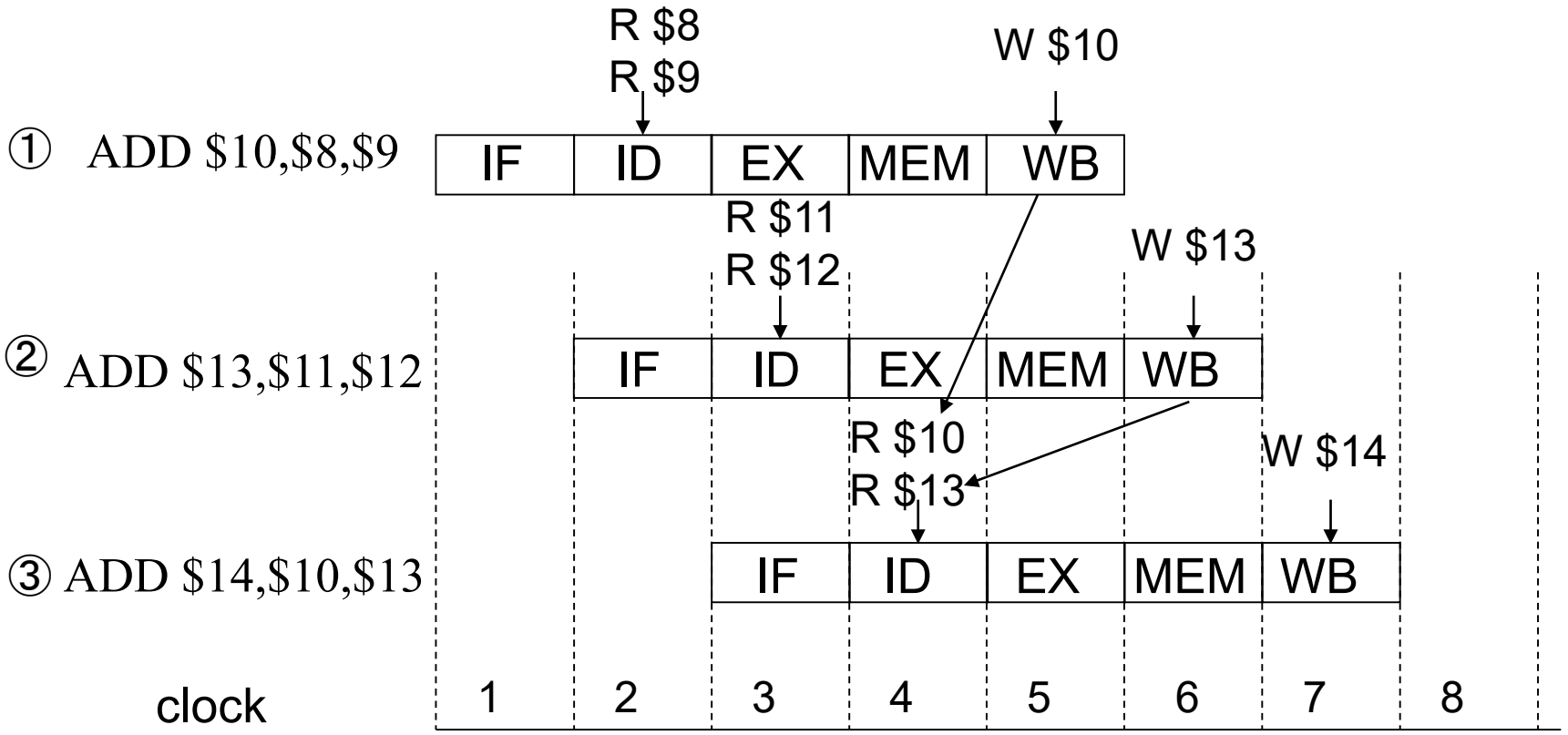
$8=1,$9=2,$11=5,$12=6,$14=7,$15=8を設定する
初期状態、および1クロック入力毎にレジスタ等を観測する

パイプライン制御

- 命令によっては、パイプラインがうまく動作しない場合がある
- データ依存
  - 先行命令の格納した値を後続命令で利用する場合
- 制御依存
  - 分岐命令の場合

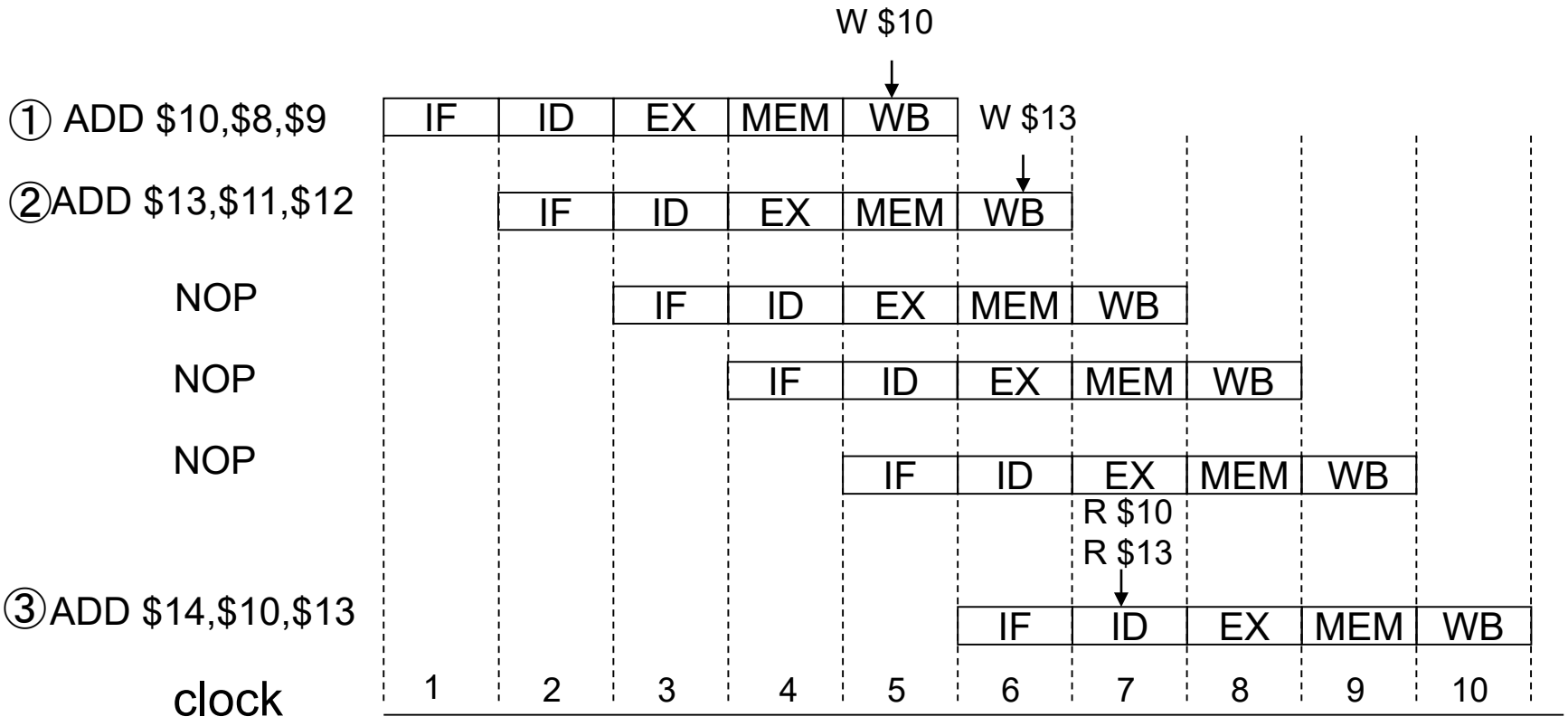
データ依存

先行命令で書き込んだ値を後続命令で参照する



命令③は正しく実行できない
（$10、$13が書き込まれる前に読み出している）

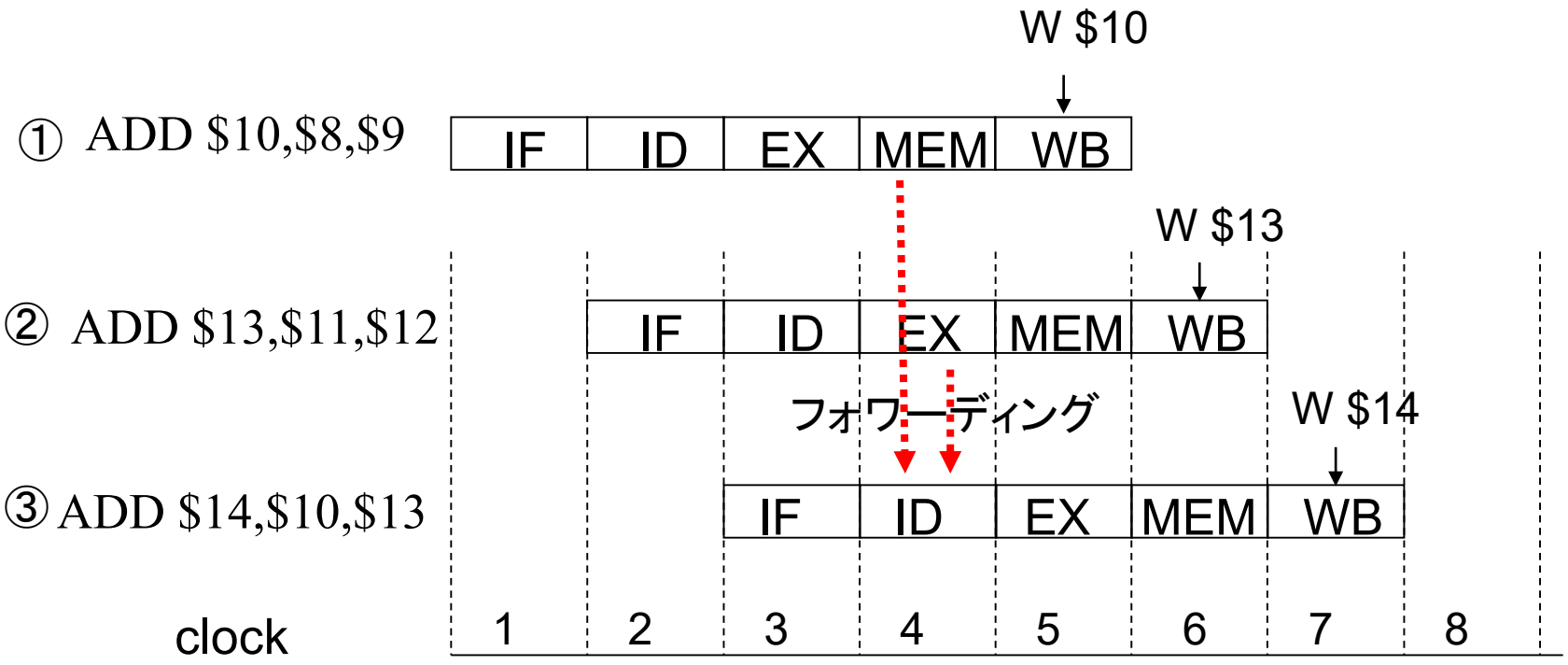
NOP命令の挿入によるデータ依存の解消



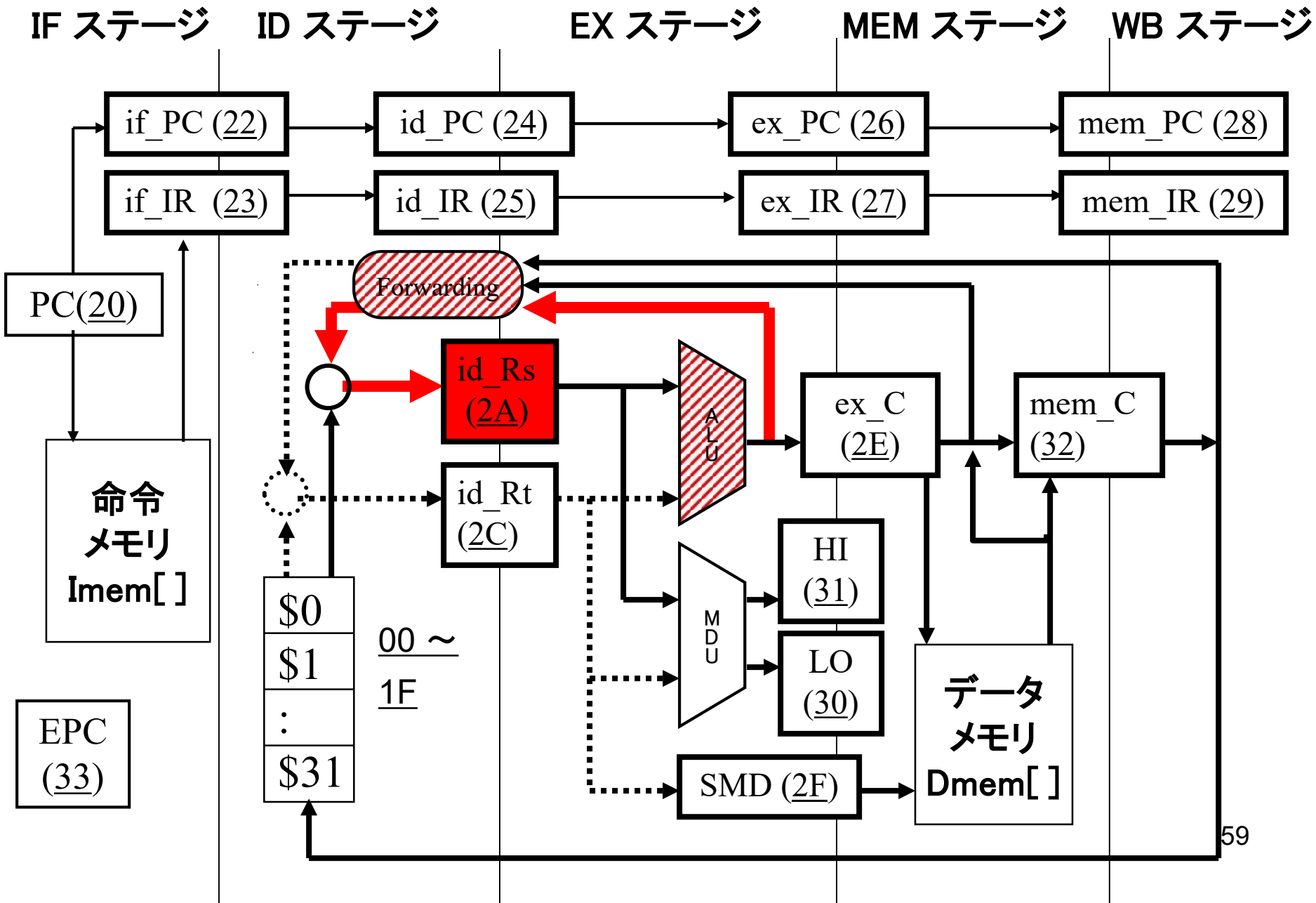
NOP命令：何もしない命令
命令③の実行が3クロック遅れ、性能が低下する

フォワーディング

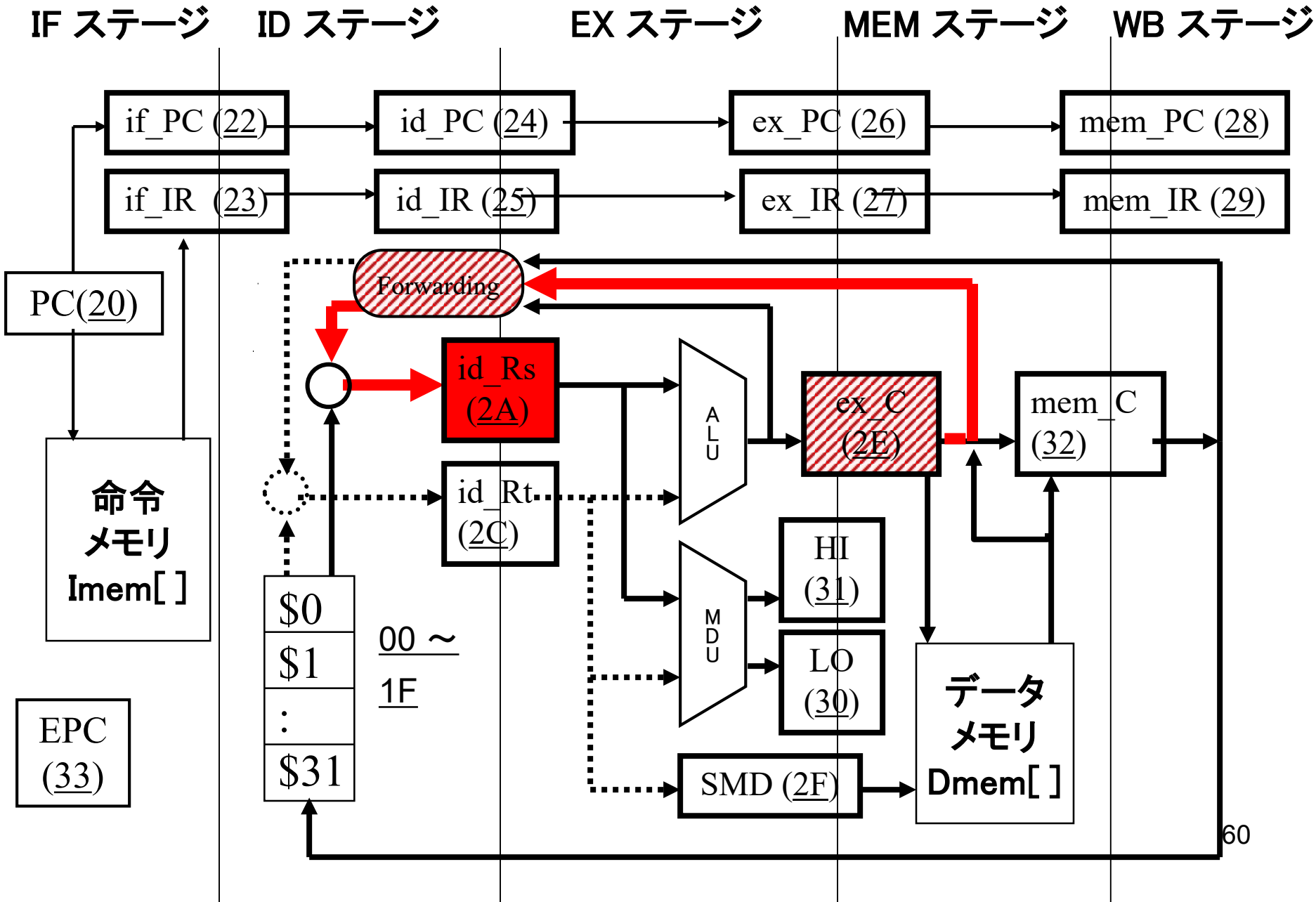
演算結果をレジスタに書き込む前に内部レジスタに格納されている値を利用する



フォワーディングの仕組み（EX→ID）



フォワーディングの仕組み(MEM→ID)

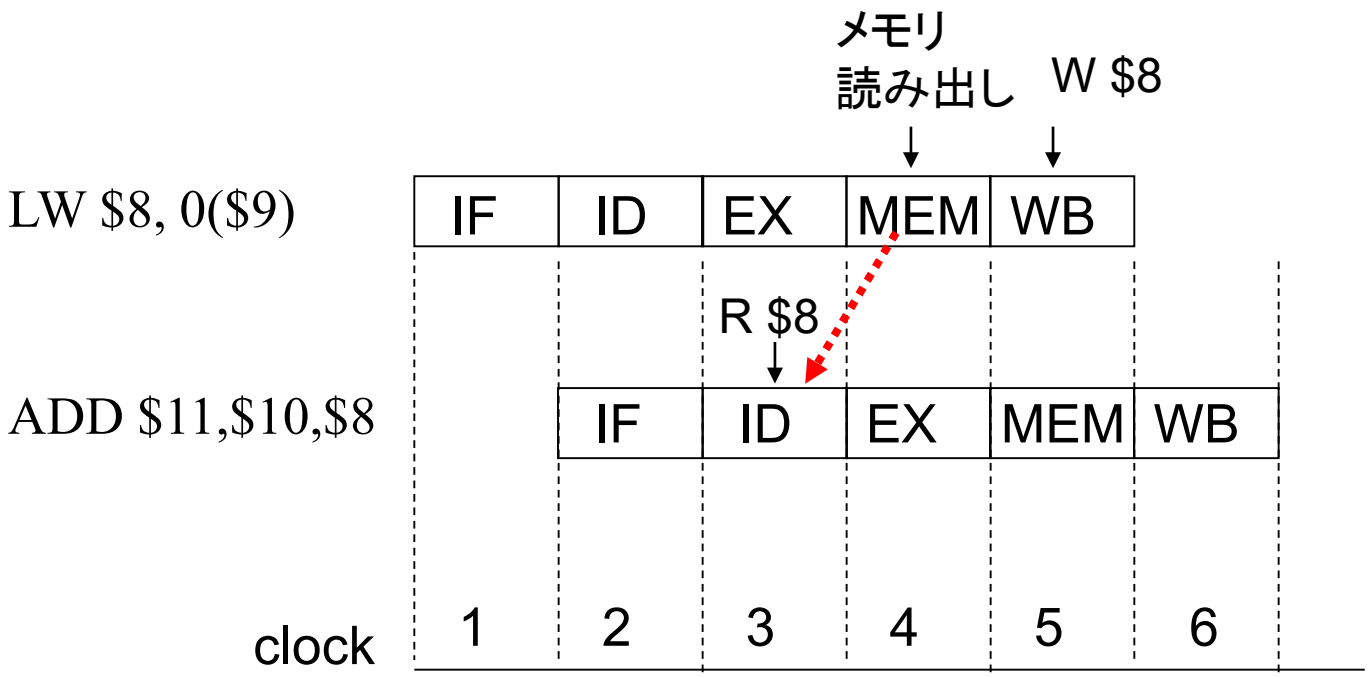


課題2-6：フォワーディングの観測

	アドレス	内容	アセンブリ言語	機能
①	8000 0000	0109 5020	ADD $10, $8, $9	$10 <= $8 + $9
②	8000 0004	016C 6820	ADD $13, $11, $12	$13 <= $11 + $12
③	8000 0008	014D 7020	ADD $14, $10, $13	$14 <= $10 + $13
④	8000 000C	0000 0000	NOP	
⑤	8000 0010	0000 0000	NOP	
⑥	8000 0014	0000 0000	NOP	
⑦	8000 0018	0000 0000	NOP	
⑧	8000 001C	0000 0000	NOP	

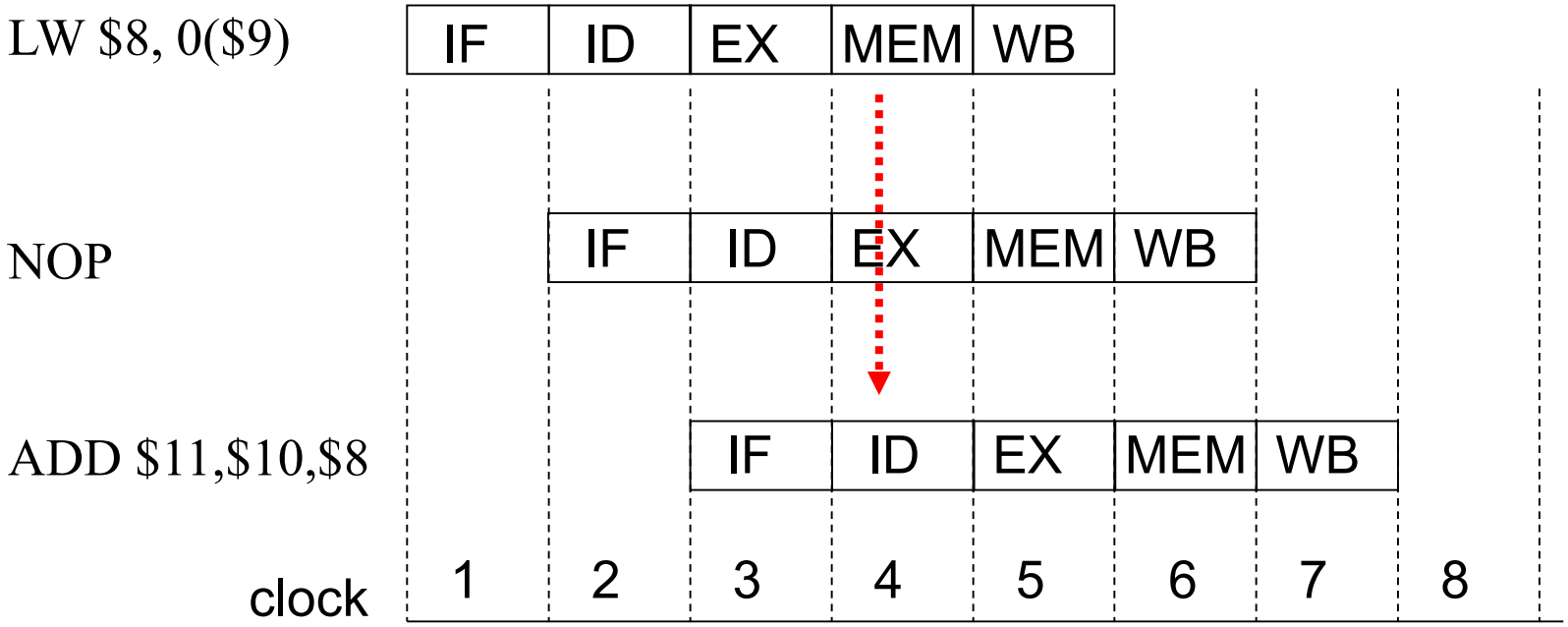
$8=9,$9=A,$11=B,$12=Cを設定し、パイプラインレジスタ等を観測することにより、フォワーディングを確認する

LW命令とデータ依存



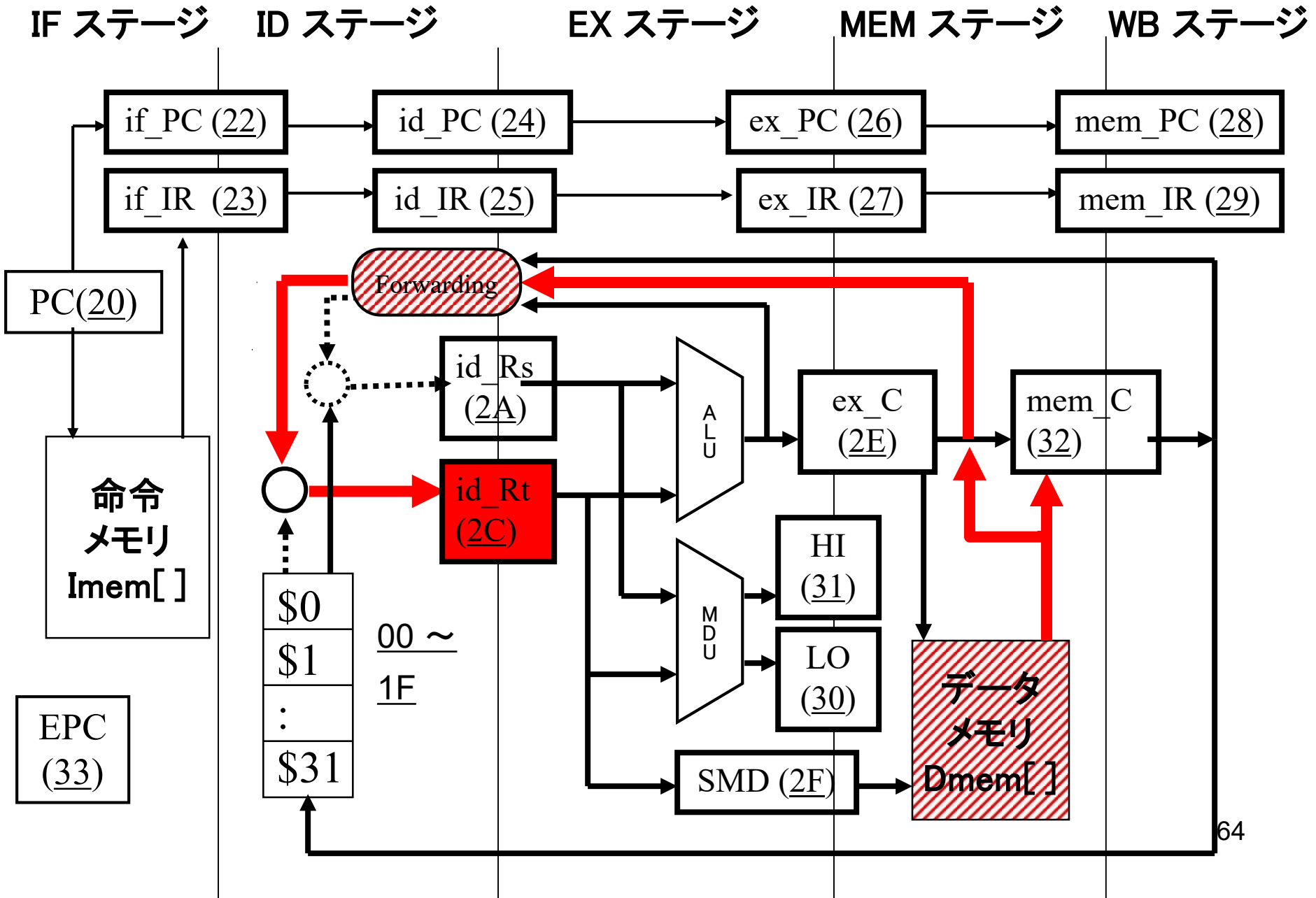
LW命令ではメモリを読み出すのはMEMステージであるので、読み出し結果を次の命令で用いる場合はフォワーディングしても間に合わない

遅延ロード



LW命令の次にNOP命令を挿入することにより、正しく実行できる

LW命令におけるフォワーディング



課題2-7：LW命令のフォワーディングの観測

	アドレス	内容	アセンブリ言語	機能
①	8000 0000	8D28 0000	LW $8, 0($9)	$8 <= MEM[$9+0]
②	8000 0004	0148 5820	ADD $11, $10, $8	$11 <= $10 + $8
③	8000 0008	0000 0000	NOP	
④	8000 000C	0000 0000	NOP	
⑤	8000 0010	0000 0000	NOP	
⑥	8000 0014	0000 0000	NOP	
⑦	8000 0018	0000 0000	NOP	

①と②の間にNOP命令を挿入した場合と、挿入しない場合を比較する。
$9=C00000000, MEM[C00000000]と$9に適切な値を設定し、
1クロック毎にレジスタ等を観測する

進捗確認

① 課題2-1についてTAによる確認
② 課題2-2についてTAによる確認
③ 課題2-3についてTAによる確認
④ 課題2-4についてTAによる確認
⑤ 課題2-5についてTAによる確認

ノルマ

⑥ 課題2-6についてTAによる確認
⑦ 課題2-7についてTAによる確認
⑧ 小テストの回答

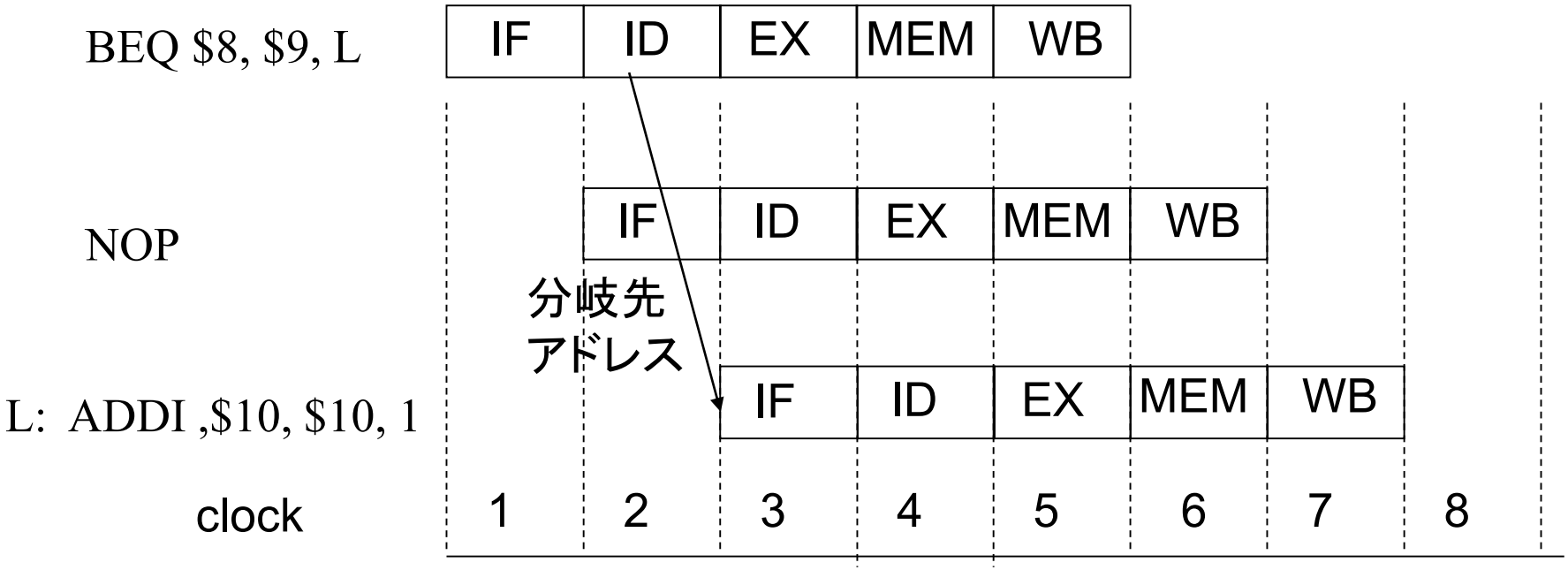
各課題において、動作内容をTAに説明すること

第3週

分岐命令のパイプライン実行

分岐先アドレスの決定がIDステージなので、分岐命令の次の命令がフェッチできない （制御依存）

遅延分岐



分岐命令は分岐の成否にかかわらず、次の命令を実行してから分岐する。分岐命令の直後にNOP命令を挿入すれば、どのような場合でも正しく分岐できる。

コンパイラ、アセンブラの役割

- 遅延ロード、遅延分岐を自動的に処理
  - NOP命令を挿入する
  - 命令の実行順序を入れ替えて最適化する
    （結果に依存しない命令をロード、分岐の直後に置く）

課題3-1：遅延分岐の観測

	アドレス	内容	アセンブリ言語	機能
①	8000 0000	1109 0001	BEQ $8, $9, L	if ($8 == $9) goto L
②	8000 0004	214A 0001	ADDI $10, $10, 1	$10 <= $10 + 1
③	8000 0008	214A 0001	L: ADDI $10, $10, 1	$10 <= $10 + 1
④	8000 000C	0000 0000	NOP	
⑤	8000 0010	0000 0000	NOP	
⑥	8000 0014	0000 0000	NOP	
⑦	8000 0018	0000 0000	NOP	

分岐成立の場合と不成立の場合の両方を観測する
$8,$9,$10に適当な値を設定する

課題3-1：遅延分岐の観測（NOP命令の挿入）

	アドレス	内容	アセンブリ言語	機能
①	8000 0000	1109 0002	BEQ $8, $9, L	if ($8 == $9) goto L
②	8000 0004	0000 0000	NOP	
③	8000 0008	214A 0001	ADDI $10, $10, 1	$10 <= $10 + 1
④	8000 000C	214A 0001	L: ADDI $10, $10, 1	$10 <= $10 + 1
⑤	8000 0010	0000 0000	NOP	
⑥	8000 0014	0000 0000	NOP	
⑦	8000 0018	0000 0000	NOP	
⑧	8000 001C	0000 0000	NOP	

分岐成立の場合と不成立の場合の両方を観測する
$8,$9,$10に適当な値を設定する

課題3-2：総和を求めるプログラムのパイプライン化

	アドレス	内容	アセンブリ言語	機能
①	8000 0000	0000 4020	ADD $8,$0,$0	$8 <= 0
②	8000 0004	3C09 C000	LUI $9,0xC000	$9 <= C000 0000
③	8000 0008	8D2A 0000	LOOP: LW $10,0($9)	$10 <= M[$9]
④	8000 000C	1140 0003	BEQ $10,$0,HALT	if ($10==0) goto HALT
⑤	8000 0010	010A 4020	ADD $8,$8,$10	$8 <= $8 + $10
⑥	8000 0014	2129 0004	ADDI $9,$9,4	$9 <= $9 + 4
⑦	8000 0018	0800 0002	J LOOP	goto 8000 0008
⑧	8000 001C	0800 0007	HALT: J HALT	halt

パイプライン実行で問題の生じる箇所を見つけよ。
また、NOP命令を挿入して問題を解決せよ。

課題3-3：総和を求めるプログラムのパイプライン実行最適化

命令の順序を入れ替えて、NOP命令の挿入箇所を減らす

課題3-4：最大値を求めるプログラムのパイプライン実行最適化

課題1-3で作成した最大値を求めるプログラムにおける命令の順序を入れ替えて、NOPの挿入箇所を減らし、最適化する

課題3-4：最大値を求めるプログラムのパイプライン実行最適化

	アドレス	内容	アセンブリ言語	機能
①	8000 0000	0000 4020	ADD $8,$0,$0	$8 <= 0
②	8000 0004	3C09 C000	LUI $9,0xC000	$9 <= C000 0000
③	8000 0008	8D2A 0000	LOOP: LW $10,0($9)	$10 <= M[$9]
④	8000 000C	1140 0005	BEQ $10,$0,HALT	if ($10==0) goto HALT
⑤	8000 0010	010A 582A	SLT $11, $8, $10	テスト
⑥	8000 0014	1160 0001	BEQ $11, $0, L	条件分岐
⑦	8000 0018	0140 4020	ADD $8,$0,$10	$8 <= $10
⑧	8000 001C	2129 0004	L: ADDI $9,$9,4	$9 <= $9 + 4
⑨	8000 0020	0800 0002	J LOOP	goto 8000 0008
⑩	8000 0024	0800 0009	HALT: J HALT	halt

進捗確認

① 課題3-1についてTAによる確認
② 課題3-2についてTAによる確認
③ 課題3-3についてTAによる確認　ノルマ
④ 課題3-4についてTAによる確認
⑤ 小テストの回答

各課題において、動作内容をTAに説明すること

第1回レポートについて

- MANABAーRに提出すること
- pdf形式
- 内容
  - それぞれの実験
    - 実験内容：実験手順、プログラムの仕様など
    - 実験結果：観測結果、プログラム、動作結果等
    - 考察：実験により気づいたこと
  - 研究課題（任意）：加点対象
  - 感想：実験全体について