提出日：2023 年 6 月 9 日

2023 年度　3 回生前期学生実験 HW

設計プロセッサユーザーズマニュアル

23 班

構成員：加藤利梓・神事倫紀・PYII PHYO MAUNG

1. 仕様・特徴

設計したプロセッサ(以下、5P-SIMPLE(Five-stage pipeline SIxteen-bit MicroProcessor for Laboratory Experiment)とよぶ)の仕様を表 1 に示す。
5P-SIMPLE は五段パイプラインプロセッサであり、命令とデータが書き込まれた.mif ファイルを読み込み、命令を順次実行することができる。

項目	仕様
最大動作可能周波数	100MHz
命令セットアーキテクチャ	項目 3 参照
コアの種類	IP コア
コア数	1

表 1　5P-SIMPLE の仕様

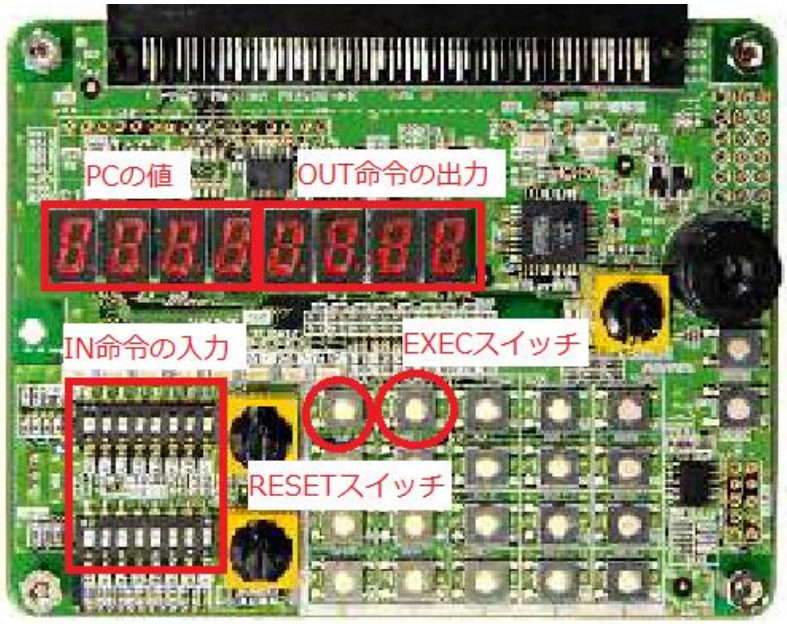

図 1　使用する PowerMedusa MU500-RX/RK 上の入出力装置への割り当て①

入出力装置	概要
PC(7SEG 左 4 ケタ)	Program Counta の値を表示する
OUT 命令の出力(7SEG 右 4 ケタ)	OUT 命令(後述)の出力(16 進数)
IN 命令の入力(ディップスイッチ)	IN 命令(後述)の入力(16 進数)
EXEC スイッチ(テンキースイッチ SW4)	起動/停止スイッチ
RESET スイッチ(テンキースイッチ SW5)	リセットスイッチ、押すとレジスタの内容を初期化する(メモリの値は初期化されない)

表 2　使用する PowerMedusa MU500-RX/RK 上の入出力装置への割り当て①

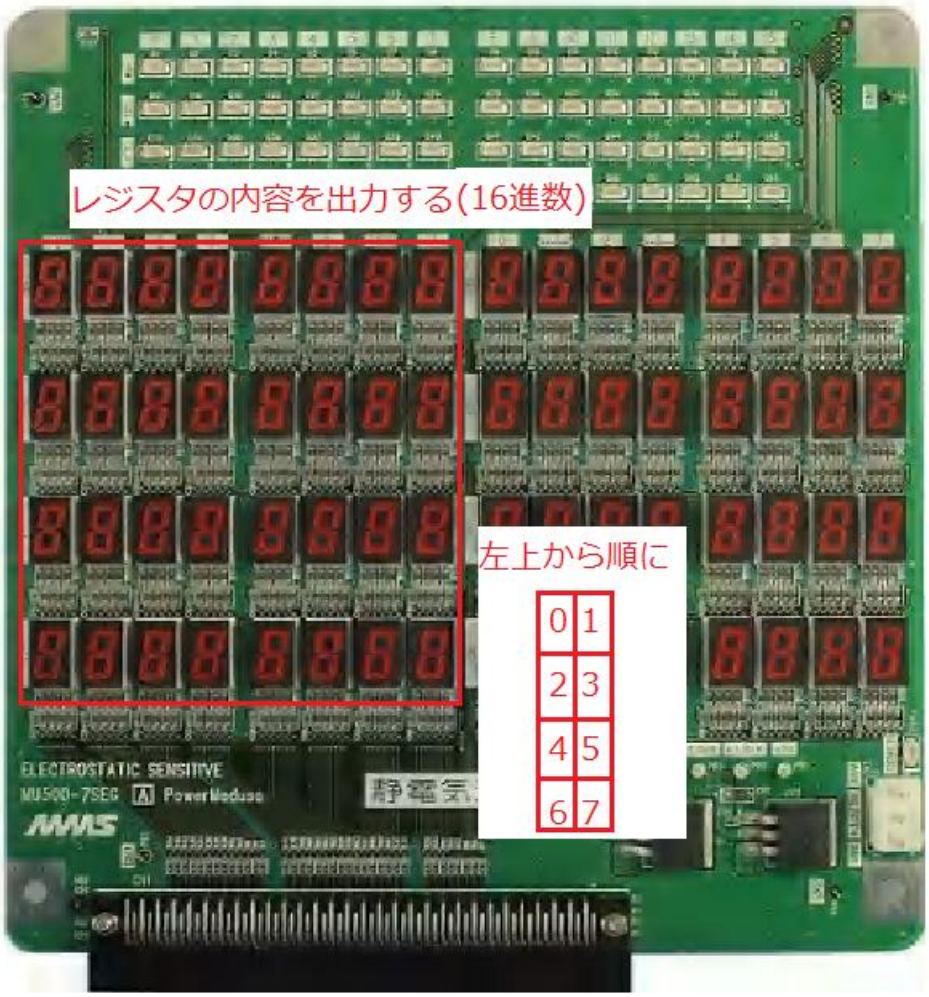

図２　使用する PowerMedusa MU500-RX/RK 上の入出力装置への割り当て②

入出力装置	概要
レジスタの内容(7SEG 左側)	レジスタの内容を全て表示する(16 進数)

表３　使用する PowerMedusa MU500-RX/RK 上の入出力装置への割り当て②

図１、図２、表２、表３に使用する FPGA ボード PowerMedusa MU500-RX/RK 上の入出力装置への割り当てを示す。

特徴

・五段パイプライン・ハーバードアーキテクチャによるマルチサイクルプロセッサーに比べて高速な命令実行

・即値加算命令(ADDI,SUBI,CMPI,SLI)による命令数の減少

→SLI(即値左論理シフト)は 0~15 の値を 0~15 ケタ左論理シフトして指定のレジスタに書き込む命令。1024 など 2 のべき乗の値をレジスタに保持したいときに用いる。

・レジスタの内容を常時確認可能により、プログラムのデバッグが容易

2. 動作方法

5P-SIMPLE を動作させる方法を示す。

5P-SIMPLE は Intel PSG の Quartus Prime Standard Edition を使って設計されており、動作させるときもこのソフトウェアを用いる。(動作させるときは Lite Edition でもよい。)

また、動作ボードは三菱電機マイコン機器ソフトウエア株式会社の FPGA ボードである PowerMedusa MU500-RX/RK を使用する。

① ダウンロードしたプロジェクト全体のファイルを適当なディレクトリに保存し、Quartus 上で開く。

② ram.v を開き、命令とデータが記録された.mif ファイルを altsyncram_component.init_file として指定する。(図 3)

```
defparam
    altsyncram_component.clock_enable_input_a = "BYPASS",
    altsyncram_component.clock_enable_output_a = "BYPASS",
    altsyncram_component.init_file = "sorted.mif",
    altsyncram_component.intended_device_family = "Cyclone IV E",
    altsyncram_component.lpm_hint = "ENABLE_RUNTIME_MOD=YES,INSTANCE_NAME=NONE",
    altsyncram_component.lpm_type = "altsyncram",
    altsyncram_component.numwords_a = 4096,
    altsyncram_component.operation_mode = "SINGLE_PORT",
    altsyncram_component.outdata_aclr_a = "NONE",
    altsyncram_component.outdata_reg_a = "UNREGISTERED",
    altsyncram_component.power_up_uninitialized = "FALSE",
    altsyncram_component.read_during_write_mode_port_a = "NEW_DATA_NO_NBE_READ",
    altsyncram_component.widthad_a = 12,
    altsyncram_component.width_a = 16,
    altsyncram_component.width_byteena_a = 1;
```

図 3　mif ファイルの指定方法

③ Tools→Programmer を開き、FPGA ボードにプロセッサをダウンロードする。
主記憶の内容を見たいときは、Tools→In-System Memory Content Editor を用いる。

④ クロック用ロータリースイッチの位置は「0」にしておく。

⑤ EXEC スイッチを押せばプログラムが作動する。(図 1)

3. 命令セットアーキテクチャ

5P-SIMPLE の命令長ははすべて 16bit の固定長とし、以下の四種類の命令形式がある。

(1) 演算/入出力命令形式

演算/入出力命令の命令セットを表 4 に示す。

- $I_{15:14}$(op1)…操作コード(11)
- $I_{13:11}$(Rs)…ソースレジスタ番号
- $I_{10:8}$(Rd)…デスティネーションレジスタ番号
- $I_{7:4}$(op3)…操作コード(0000~1111)
- $I_{3:0}$(d)…シフト桁数

mnemonic	op3	function
ADD	0000	r[Rd] = r[Rd] + r[Rs]
SUB	0001	r[Rd] = r[Rd] - r[Rs]
AND	0010	r[Rd] = r[Rd] & r[Rs]
OR	0011	r[Rd] = r[Rd] | r[Rs]
XOR	0100	r[Rd] = r[Rd] ^ r[Rs]
CMP	0101	r[Rd] - r[Rs]
MOV	0110	r[Rd] = r[Rs]
reserved	0111	
SLL	1000	r[Rd] = shift_left_logical(r[Rd],d)
SLR	1001	r[Rd] = shift_left_rotate(r[Rd],d)
SRL	1010	r[Rd] = shift_right_logical(r[Rd],d)
SRA	1011	r[Rd] = shift_left_arithmetic(r[Rd],d)
IN	1100	r[Rd] = input
OUT	1101	output = r[Rs]
reserved	1110	
HLT	1111	停止

表 4　演算/入出力命令

(2) ロード/ストア命令形式

ロード/ストア命令の命令セットを表 5 に示す。

- $I_{15:14}$(op1)…操作コード(00/01)
- $I_{13:11}$(Ra)…ソース/デスティネーションレジスタ番号
- $I_{10:8}$(Rb)…ベースレジスタ番号
- $I_{7:0}$(d)…変位

mnemonic	op1	function
LD	00	r[Ra] = *(r[Rb] + sign_ext(d))
ST	01	*(r[Rb] + sign_ext(d)) = r[Ra]

表 5　ロード/ストア命令

(3) 即値ロード/即値演算/即値シフト/無条件分岐命令形式

即値ロード/即値加算/即値シフト/無条件分岐命令の命令セットを表 6 に示す。

- $I_{15:14}$(op1)…操作コード(10)
- $I_{13:11}$(op2)…操作コード(000~110)
- $I_{10:8}$(Rb)…ソース/デスティネーション/ベースレジスタ番号
- $I_{7:0}$(d)…即値または変位

mnemonic	op2	function
LI	000	r[Rb] = sign_ext(d)
ADDI	001	r[Rb] = r[Rd] + sign_ext(d)
SUBI	010	r[Rb] = r[Rd] - sign_ext(d)
CMPI	011	r[Rb] - sign_ext(d)
B	100	PC = PC + 1 + sign_ext(d)
SLI	101	r[Rb] = shift_left_logical(d[7:4],d[3:0])
reserved	110	
条件分岐命令	111	

表 6　即値ロード/即値演算/即値シフト/無条件分岐命令

(4) 条件分岐命令形式

条件分岐命令の命令セットを表 7 に示す。

- $I_{15:14}$(op1)…操作コード(10)
- $I_{13:11}$(op2)…操作コード(111)
- $I_{10:8}$(cond)…分岐条件
- $I_{7:0}$(d)…変位

mnemonic	op3	function
BE	0000	if (Z) PC = PC + 1 + sign_ext(d)
BLT	0001	if (S ^ V) PC = PC + 1 + sign_ext(d)
BLE	0010	if (Z || (S ^ V)) PC = PC + 1 + sign_ext(d)
BNE	0011	if (!Z) PC = PC + 1 + sign_ext(d)
reserved	0100	
reserved	0101	
reserved	0110	
reserved	0111	

表 7　条件分岐命令

4. 構造

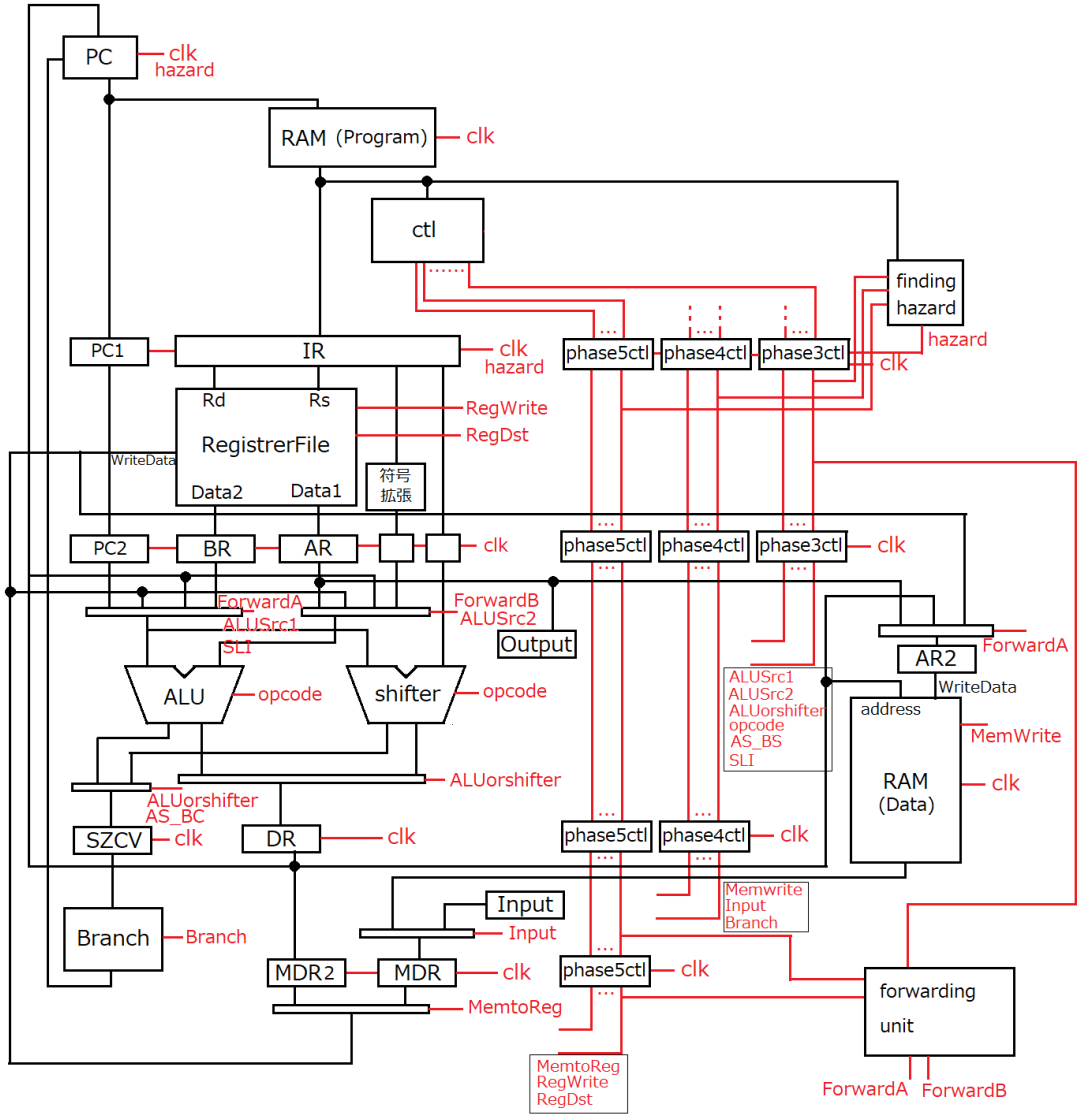

図 4　5P-SIMPLE の構造

図 4 に 5P-SIMPLE のブロック図を示す。制御信号は赤で描かれている。

図で省略している点

① pll を使って 40MHz の入力クロックを 100MHz とし、各種コンポーネントの入力としている。

② 不成立分岐予測をしているので、分岐が成立した場合、制御信号を保持しているレジスタをフラッシュする信号がある。

コンポーネント	説明
PC	PC の値を計算し、メモリのアドレス入力に入力する。内部で+1 した値をあらかじめ計算しておき、分岐命令が来たときは分岐先のアドレスと命令と分岐するかどうかの信号を受けとり、次のアドレスを出力する。
ctl	命令を受け取り、各制御信号を出力する。
各種データ用レジスタ (IR,AR,AR2,BR,PC1,PC2, SZCV,DR,MDR,MDR2)	フェーズごとにデータを受け取り、保持し、出力する。
各種制御信号用レジスタ (phase3ctl,phase4ctl, phase5ctl)	フェーズごとに制御信号を受け取り、保持し、出力する。 phase *n* ctl はフェーズ n で使われる信号を保持する。
RegisterFile	16bit 8 本のレジスタから構成されている。レジスタの読み出しと書き込みをする。
ALU	四則演算、論理演算、移動演算、比較演算を計算する。
Shifter	シフト演算を計算する。
Branch	条件コードと branch 命令の種類を示す信号受け取り、分岐するかどうかの信号を出力する。
RAM(Program)	命令用の主記憶メモリ。命令をここから読み出す。
RAM(Data)	データ用の主記憶メモリ。データの読み出しと書き込みをする。
finding_hazard	ハザードを検出し、hazard 信号を出す。
forwarding_unit	フォワーディングが必要な場合を検知し、forward 信号を出す。
pll	40MHz のクロックを入力として、100MHz のクロックを出力する。

表5　各コンポーネントの説明

表5に各コンポーネントの説明を示す。

5. 動作

5.1. 制御回路

以下の信号が外部から供給される。

1. クロック

① 20MHz クロック：7SEG のダイナミック点灯に用いる。

② 40MHz クロック：pll の入力とし、100MHz のクロックに変換する。

2. reset…リセット信号、この信号が 1 になると内部コンポーネントをリセットする。
3. exec…起動/停止信号、停止状態の時に 1 になると命令の実行を開始し、実行状態の時に 0 になるとその時点で実行中の命令を完了してから停止する。

5.2. フェーズ

命令実行を 5 つのフェーズに分け、五段パイプラインを実現している。

フェーズ①：命令用メモリから命令のフェッチをし、IR に格納、ctl に入力し、制御信号用レジスタに制御信号を格納する。

フェーズ②： IR が保持する命令の Ra/Rs フィールドと Rb/Rd フィールドで指定される汎用レジスタの値を読み出し、AR と BR に保持する。

フェーズ③：ALU またはシフターで命令が定める演算を行い、その結果を DR に保持する。また、条件コードを SZCV に保持する。条件コードが更新されるのは演算命令の時のみである。Output 命令の時は値を出力する。

フェーズ④：ロードストア命令において、DR の値をアドレスとしてデータ用メモリにアクセスする。ロード命令の時は値を読み出し MDR に格納する。ストア命令の時は AR2 が保持する内容を書き込む。また、Input 命令では、入力スイッチの値を MDR に格納する。また、DR の値を MDR2 に格納する。
また、分岐命令の分岐が成立するか否かの判断を行う。

フェーズ⑤：汎用レジスタへの書き込みを伴う命令で、DR または MDR の値を命令の Ra,Rb または Rs フィールドで指定される汎用レジスタに書きこむ。また、分岐命令で分岐が成立するときは DR の値を分岐先アドレスとして PC に書き込む。このときに制御信号を保持しているレジスタをフラッシュする

5.3. 主記憶

5P-SIMPLE はハーバードアーキテクチャを採用しているので、命令を読み出す用のメモリとデータ用のメモリを分けて実装している。それぞれの主記憶の容量は 4096words=4KW である。

5.4. フォワーディング

5P-SIMPLE はパイプラインプロセッサなので、フォワーディングが必要になる場合がある。次の 2 通りである。

① 一つ次の第 5 フェーズで書き込もうとしているレジスタの値を一つ次の第 3 フェーズの演算で用いる場合。

② 二つ次の第 5 フェーズで書き込もうとしているレジスタの値を一つ次の第 3 フェーズの演算で用いる場合。

それぞれの対応は以下のとおりである。

① 一つ次の第 5 フェーズで書き込もうとしているレジスタの値をフォワーディングして一つ次の第 3 フェーズで使えるようにする。(図 5)

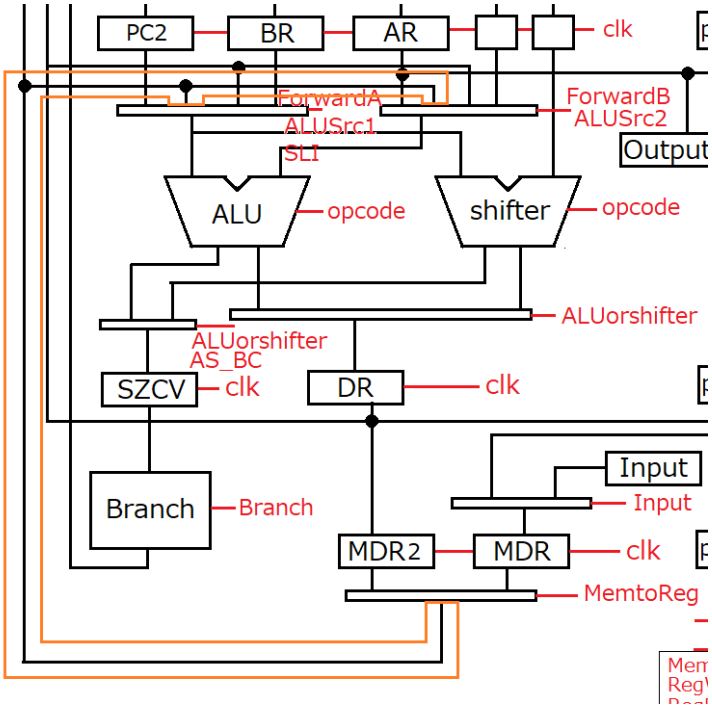

図 5　フォワーディング回路①

② 二つ次の第５フェーズで書き込もうとしているレジスタの値をフォワーディングして一つ次の第３フェーズで使えるようにする。（図６）

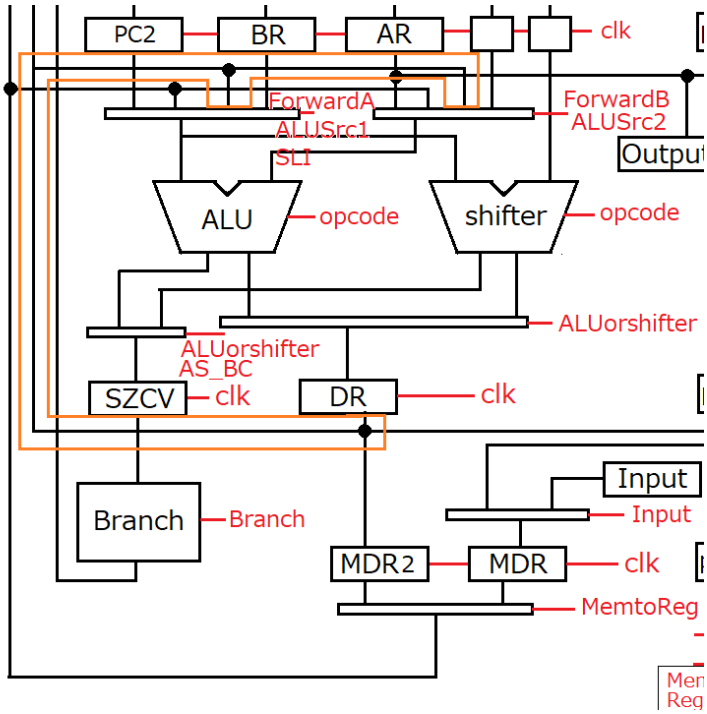

図６　フォワーディング回路②

このフォワーディングを実現するためのコンポーネントが forwarding_unit である。

5.5. ハザード検出

前述のフォワーディングでも対処ができない場合がある。それはロード命令または Input 命令で読みだし、レジスタに格納する値を次の演算命令で使う場合である。この場合はフォワーディングでも対処ができないので、命令の実行を一つ遅らせる必要がある。

ハザード検出は第１フェーズと第２フェーズにまたがって行われる。すなわち、第２フェーズにいる命令がロード命令または Input 命令の時で、かつ第１フェーズにいる命令がロード命令または Input 命令で読みだし、レジスタに格納する値を使用するときにハザードが検出される。

ハザードが検出されると PC のカウントをストップし、今第 1 フェーズにいる命令を nop 命令に書き換える。

このハザード検出を実現するためのコンポーネントが finding_hazard である。