# 平成 29 年度　計算機科学実験及演習 3　　HW

# 中間報告

# 機能設計仕様書

提出日
5 月 11 日

グループ 22
1029277526　白石竜也

# 1　分割

プロセッサは図 1.1 のようなコンポーネントに分割した。各フェーズに相当する IF、ID、EX、MA、WB をそれぞれ 1 つのモジュールとして設計する。この 5 つのコンポーネントのうち自分は IF、ID を担当する。

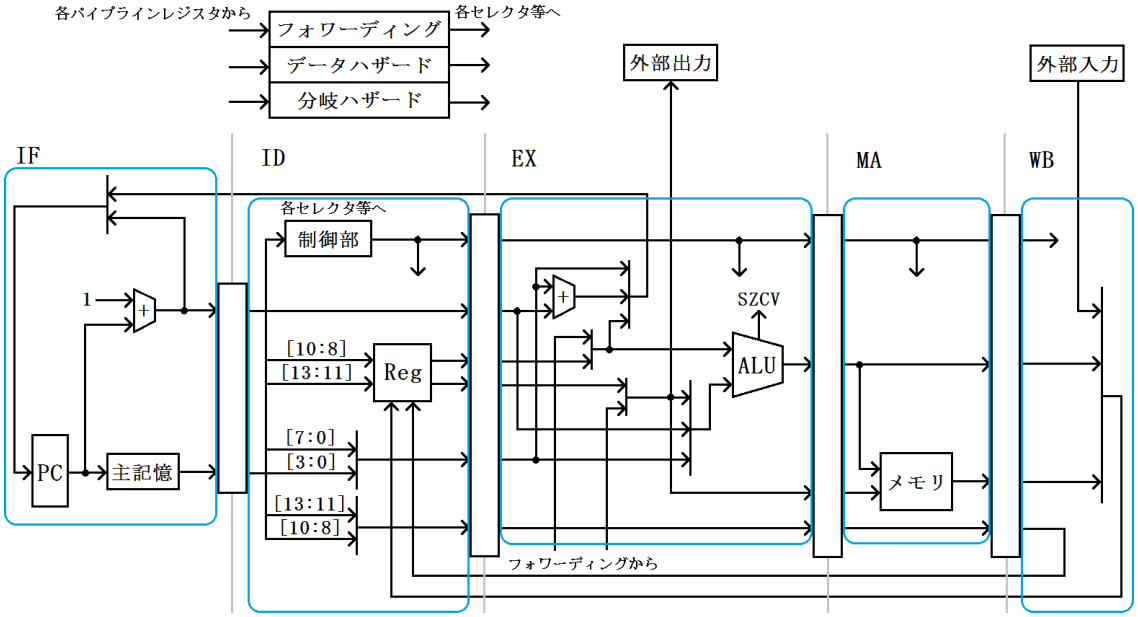


図 1.1　プロセッサの分割

# 2　IF

## 2.1　外部仕様

IF ではクロック信号の立ち上がりに応じて、主記憶から命令を取り出す。命令はアドレス 0x0 のものから順に取り出される。出力はすべて次の ID ステージに渡される。入出力は以下の通りである。

### 入力

clock　クロック信号
reset　リセット信号 (負論理)
stop　停止信号
Branch　分岐信号
PC_branch[15:0]　分岐先アドレス

出力

PC_plus1[15:0]　PC+1
instruction[15:0]　命令

入力信号の説明をする。リセット信号 reset は負論理で、0 にすると出力がすべて 0 にセットされ、次の命令の取り出しは 0x0 から始まる。停止信号 stop を 1 にすると、その時点で出力している命令を取り出し続ける。分岐信号 Branch を 1 にすると、次の命令の取り出しが PC _ branch で指定されたアドレスから始まる。

出力信号の説明をする。PC _ plus1 はその時点で出力している命令のアドレス +1 の値である。instruction からは取り出された命令が出力される。

## 2.2　内部仕様

IF のブロック図を図 2.1 に示す。外部仕様にも示したように、IF は主記憶から命令を取り出すのが主な役割である。

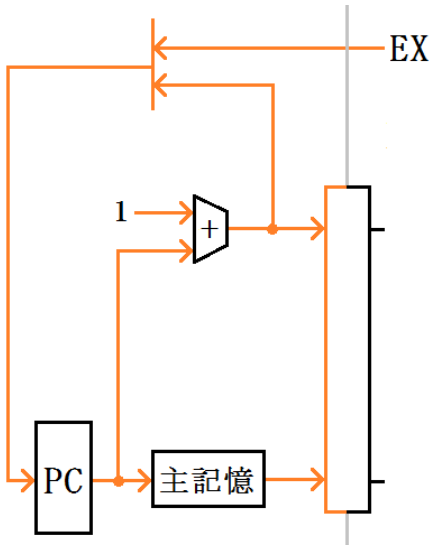


図 2.1　IF

設計で留意すべき点は PC の実装と、分岐が反映されるタイミングである。PC をブロック図の位置通りにレジスタとして置いたときの動作を図 2.2 に示す。クロックの立ち上がりとともに、主記憶はその瞬間入力されていた PC の値 (青) をアドレスに命令を取り出す。そして PC は新しい値 (緑) に更新される。ここで EX から分岐先アドレス (橙) に分岐するように信号が来たとすると左図のようになる。次のクロックで分岐先アドレスの命令を取り出して欲しいのだが、PC を挟んでいるため右図のようになりワンテンポ遅れてしまう。

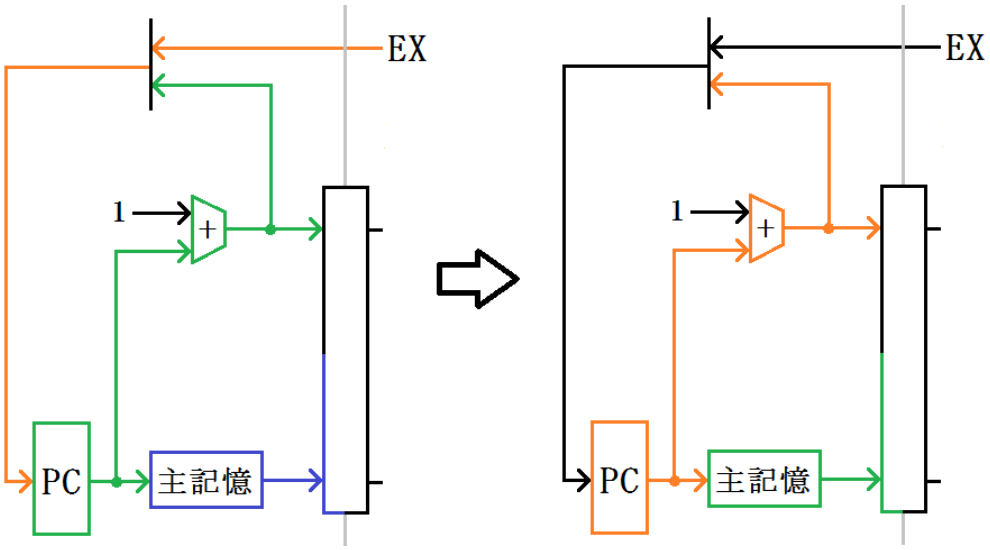


図 2.2　IF の留意点

そこで実際の回路上は、図 2.3 のように IF・ID 間のパイプラインレジスタを PC の代わり (実際には PC+1 が格納されている) にするような構造になっている。このようにすると、分岐を次のクロックで再現することができる。

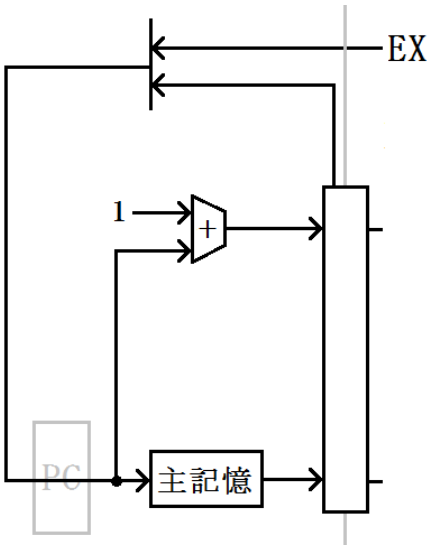


図 2.3　IF の実際

# 3 ID

## 3.1 外部仕様

ID ではクロック信号の立ち上がりに応じて、レジスタの読み書きを行う。そして命令に応じたレジスタの値、ビット拡張した即値、書き込みアドレスを出力する。また、後続ステージの制御信号も生成する。入出力は以下の通りである。

入力

clock　クロック信号
reset　リセット信号 (負論理)
stop　停止信号
PC_in[15:0]　PC+1
instruction[15:0]　命令
RegWrite　書き込み信号
WBaddress_in[2:0]　書き込みアドレス
WBdata[15:0]　書き込みデータ

出力

PC_out[15:0]　PC+1
Rd_Rb[15:0]　レジスタの値
Rs_Ra[15:0]　レジスタの値
immediate[15:0]　即値
WBaddress_out[15:0]　書き込みアドレス
control[ :0]　制御信号
ALUcontrol[3:0]　ALU 用制御信号

入力信号の説明をする。リセット信号 reset は IF と同様に負論理で、0 にすると出力がすべて 0 にセットされる。停止信号 stop は 1 にすると出力が固定される。PC_in、instruction は IF から送られる PC+1 の値と命令である。RegWrite、WBaddress_in、WBdata は WB から送られるデータである。レジスタへの書き込みを行う場合、RegWrite に 1 を入力すると、WBaddress_in で指定されるアドレスのレジスタに WBdata で指定されるデータが書き込まれる。

出力信号の説明をする。PC_out は入力の PC_in がそのまま出力される。Rd_Rb、Rs_Ra はレジスタの値であり、Rd_Rb からは r[Rd] または r[Rb]、Rs_Ra からは r[Rs] または r[Ra] が出力される。immediate はゼロ拡張または符号拡張した即値である。WBaddress_out は書き込みアドレスである。これは各ステージを経由した後、WB からの入力 WBaddress_in として入力される。

control は制御信号である。これの各ビットについての説明を図 3.1 に示す。ALUcontrol は ALU 用の制御信号である。これに対する ALU の動作は図 3.2 の通りである。

| | 信号名 | 説明 | | ステージ |
|---|---|---|---|---|
| control[5] | ALUSrc | 演算オペランドの選択 | 0→Rs<br>1→d | EX |
| control[4] | Branch | 分岐命令か否か | 1→分岐命令 | |
| control[3] | MemRead | メモリを読むか否か | 1→メモリを読む | MA |
| control[2] | MemWrite | メモリに書くか否か | 1→メモリに書く | |
| control[1] | RegWrite | レジスタに書くか否か | 1→レジスタに書く | WB |
| control[0] | MemtoReg | 書き込みデータの選択 | 0→演算結果<br>1→メモリのデータ | |

図 3.1　control

| | 動作 |
|---|---|
| 0000 | 加算 |
| 0001 | 減算 |
| 0010 | 論理積 |
| 0011 | 論理和 |
| 0100 | 排他的論理和 |
| 0101 | 減算 |
| 0110 | Rs を出力 |
| 0111 | 未定義 |
| 1000 | 左論理シフト |
| 1001 | 左循環シフト |
| 1010 | 右論理シフト |
| 1011 | 右算術シフト |
| 1100 | 未定義 |
| 1101 | 未定義 |
| 1110 | 未定義 |
| 1111 | 未定義 |

図 3.2　ALUcontrol

## 3.2　内部仕様

ID のブロック図を図 3.3 に示す。外部仕様にも示したように、ID は命令に応じてレジスタの値、即値、制御信号等を揃える役割を担う。

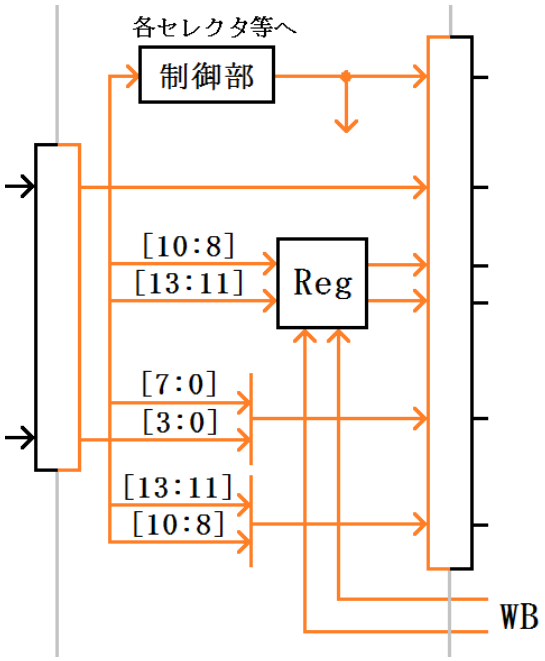


図 3.3　ID

制御信号は後続ステージで使用するものだけでなく、ID 内で使用するものも生成する。内部で使用する制御線は 2 つある。1 つは即値のビット拡張方法を選択する信号線 (Imm とする) である。例えば SLL 命令では命令の下位 4 ビットを 0 拡張し、B 命令では命令の下位 8 ビットを符号拡張する必要がある。ここでは 0 なら前者、1 なら後者を選択するように実装している。もう 1 つは書き込みアドレスを選択する信号線 (RegDst とする) である。例えば ADD 命令では Rd(10〜8 ビット目)、LD 命令では Ra(13〜11 ビット目) を選択する必要がある。ここでは 0 なら前者、1 なら後者を選択するように実装している。図 3.4 に各命令に対する内部、外部への制御線を具体的にどう設定しているか示す。*は 0 と 1 どちらでもよいことを表す。

| | Imm | RegDst | control[5:0] | | | | | | ALUcontrol[3:0] |
|---|---|---|---|---|---|---|---|---|---|
| ADD | * | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0000 |
| SUB | * | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0001 |
| AND | * | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0010 |
| OR | * | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0011 |
| XOR | * | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0100 |
| CMP | * | * | 0 | 0 | 0 | 0 | 0 | 0 | 0101 |
| MOV | * | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0110 |
| SLL | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 0 | 1000 |
| SLR | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 0 | 1001 |
| SRL | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 0 | 1010 |
| SRA | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 0 | 1011 |
| IN | * | 0 | * | 0 | 0 | 0 | 1 | 0 | 1100 |
| OUT | * | * | 0 | 0 | 0 | 0 | 0 | 0 | 1101 |
| HLT | * | * | * | 0 | 0 | 0 | 0 | 0 | 1111 |
| LD | 1 | 1 | 1 | 0 | 1 | 0 | 1 | 1 | 0000 |
| ST | 1 | * | 1 | 0 | 0 | 1 | 0 | 0 | 0000 |
| LI | 1 | 0 | 1 | 0 | 0 | 0 | 1 | 0 | 0110 |
| B | 1 | * | 1 | 1 | 0 | 0 | 0 | 0 | * |
| BE | 1 | * | 1 | 1 | 0 | 0 | 0 | 0 | * |
| BLT | 1 | * | 1 | 1 | 0 | 0 | 0 | 0 | * |
| BLE | 1 | * | 1 | 1 | 0 | 0 | 0 | 0 | * |
| BNE | 1 | * | 1 | 1 | 0 | 0 | 0 | 0 | * |

図 3.4　制御線

設計で留意すべき点はレジスタ書き込み→レジスタ読み込みの順で実行することである。これにより、パイプラインで動作したときに ID と WB 間のデータハザードをなくすことができる。