

計算機科学実験 3 A

ユーザーズマニュアル

チーム 4 : 竹田原俊介 Chung Mung Tim

2022-06-02

第1章 概要

この実験の目的は、SIMPLE (SIxteen-bit MicroProcessor for Laboratory Experiment) と呼ばれるコンピュータを設計することである。本レポートでは、SIMPLE の命令セットアーキテクチャの詳細と、各命令に対する私たちが設計した CPU 全体としての振る舞いを紹介する。

第2章 使用説明書

この章では、IP コア (ソフトコアプロセッサ) として提供することを想定し、わたしたちが設計したプロセッサを使用する上で必要十分な情報を記載する。一般に、ユーザーが実行したい 16 ビット命令をボードの主記憶に入力すると、命令を実行することができる。命令の書き方については、「第4章 命令セットアーキテクチャ」を参照してください。ここでは、ユーザーがボードに直接に触れる方法について説明する。

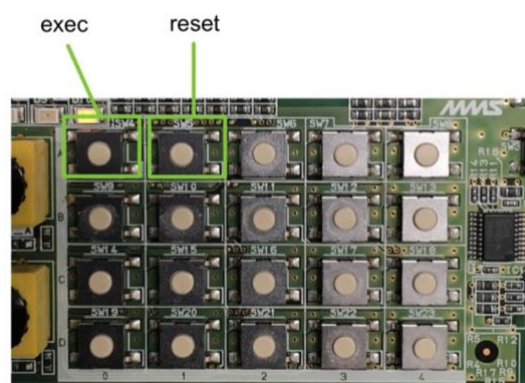
reset ボタン

リセットボタンを押せば、プロセッサは命令の実行を最初からやり直す。

exec ボタン

exec ボタンには 2 つの機能がある。停止状態にある時に押すと、命令の実行を開始する。SIMPLE が以前に停止していた場合は、停止したところから開始される。SIMPLE が実行状態にある時に押すと、その時点で実行中の命令を完了してかた停止する。reset ボタンと exec ボタンの場所については図 2.1 を参照してください。

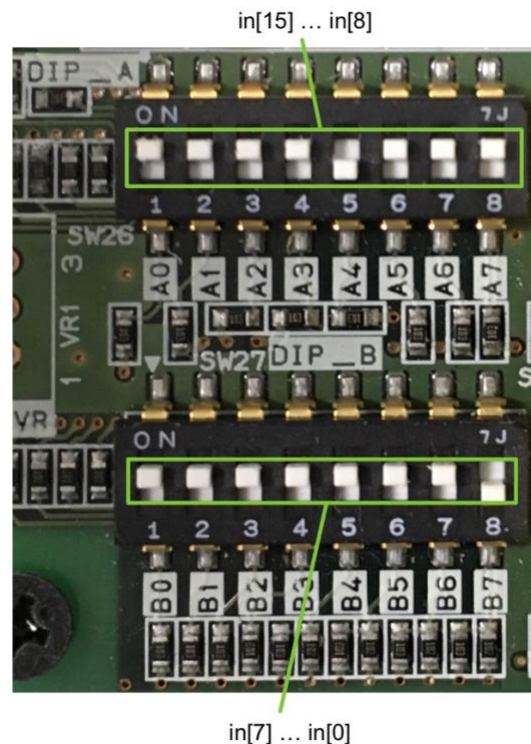
図 2.1 reset ボタンと exec ボタン



IN 命令

提供している命令の 1 つは入力 IN 命令で、ユーザーはボードから 16 桁の 2 進数を入力し、それをレジスタに保存することができる。FPGA ボードからの入力はディップスイッチを使って入力する。16 ビットの入力を図 2.2 のように桁を対応して入力することができる。ただし、ディップスイッチは負論理を使用している。

図 2.2 ボードからの入力



OUT 命令

また、レジスタに格納された 16 ビットの値を二つボードに LED などとして出力する OUT 命令も提供されている。私が設計した SIMPLE では、その二つの 16 ビットの値をそれぞれ 4 桁 16 進数として表 2.3 のように Power Medusa EC6S ボードの 8 桁 7SEG LED に映ることができる。また、命令中の 3 ビットフィールド (d) を使って、出力の場所を指定できるように設計した (図 2.4 参照)。

表 2.3 7SEG LED で 16 進数の表示

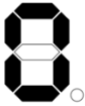




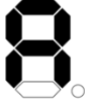







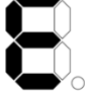

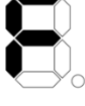
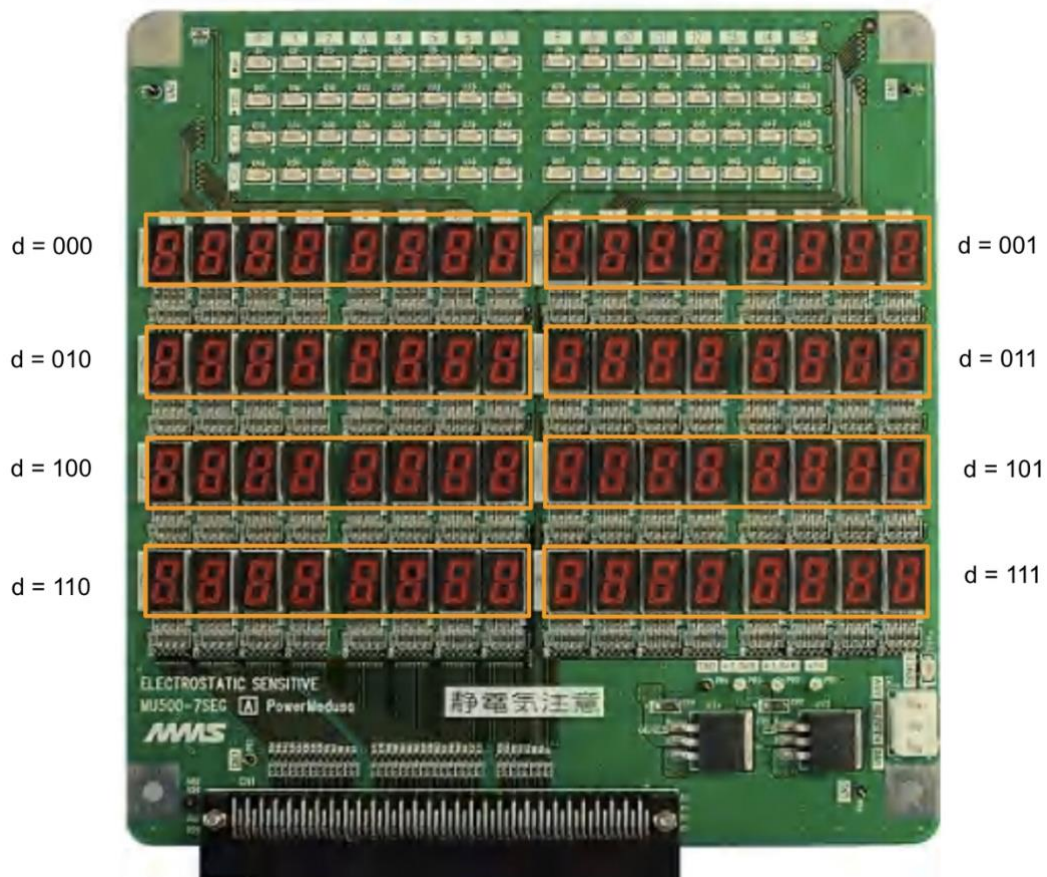
16 進数	7SEG LED 表示	16 進数	7SEG LED 表示
0		8	
1		9	
2		A	
3		B	
4		C	
5		D	
6		E	
7		F	

図 2.4 出力 OUT 命令



第3章 性能と特長

プロセッサ性能は以下である：

- サイクル数： 命令数の 5 倍
- 周波数： (最大周波数) 5MHz
- 面積 (Logical elements)： 1152/28,848 (4%)

3.2 特長

- ALU とシフタを一体化すること
- プログラムカウンタと加算器を一体化すること
- 汎用レジスタは書き込みイネブラーを持つこと
- レジスタは出力イネブラーを持つこと
- 全てのレジスタは同じクロックを共用していること
- 即値オペランドが強化されていること
- 出力命令が強化されていること

第4章 命令セットアーキテクチャ

SIMPLE の命令は全て 1 語 16 ビットの固定長である．SIMPLE は以下 4 種類の命令形式がある．その形式と機能については，4.1 節から 4.4 節で紹介する．

4.1 演算／入出力命令

15	14	13	11	10	8	7	4	3	2	0
11	Rs	Rd	op3	f	d					

- $I_{15:14}$ (11) 操作コード
- $I_{13:11}$ (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 (f) オペランドフラグ
- $I_{2:0}$ (d) シフト桁数

ただし、

- f が 0 の場合， $r[Rd]$ と $r[Rs]$ をオペランドとして使用して計算する．
- f が 1 の場合， $r[Rd]$ と $\text{sign_ext}(\text{conc}(Rs,d))$ をオペランドとして使って計算する．ここで， $\text{conc}(Rs,d)$ は Rs フィールドの 3 ビットと d フィールドの 3 ビットを結合した値を表す． $\text{sign_ext}(\text{conc}(Rs,d))$ のようにして得られるオペランドを $\text{sw}(f, Rs, d)$ と表記する

4.1.1 演算命令

SIMPLE の演算命令を表 2.1.1 に示す．演算命令では結果に基づく条件コード S, Z, C, V が設定される．

1. 算術演算

レジスタ Rd と Rs の加算 (ADD) または減算 (SUB) の結果を Rd に格納し，条件コードを設定する．

2. 論理演算

レジスタ Rd と Rs の，ビットごとの論理積 (AND)，論理和 (OR)，または排他的論理和 (XOR) の結果を Rd に格納し，条件コードを設定する．

3. 比較演算 (CMP)

レジスタ **Rd** から **Rs** を減算し、結果に基づく条件コード設定のみを行う。

4. 移動演算 (MOV)

レジスタ **Rd** に **Rs** の値を単に格納し、**Rd** の値に基づき条件コードを設定する。

5. シフト演算

レジスタ **Rd** の値を、以下のようにシフトした値を **Rd** に格納し、条件コードを設定する。

- SLL (shift left logical)

左論理シフト。左シフト後、空いた部分に 0 を入れる

- SLR (shift left rotate)

左循環シフト。左シフト後、空いた部分にシフトアウトされたビット列を入れる。

- SRL (shift right logical)

右論理シフト・右シフト後、空いた部分に 0 を入れる。

- SRA (shift right arithmetic)

右算術シフト・右シフト後、空いた部分に符号ビットの値を入れる。

シフト桁数は即値 **d** (0 ~ 15) である。

表 4.1.1 : SIMPLE の演算命令

mnemonic		op3	function
ADD	Rd, Rs	0000	$r[Rd] = r[Rd] + sw(f, Rs, d)$
SUB	Rd, Rs	0001	$r[Rd] = r[Rd] - sw(f, Rs, d)$
AND	Rd, Rs	0010	$r[Rd] = r[Rd] \& sw(f, Rs, d)$
OR	Rd, Rs	0011	$r[Rd] = r[Rd] sw(f, Rs, d)$
XOR	Rd, Rs	0100	$r[Rd] = r[Rd] \wedge sw(f, Rs, d)$
CMP	Rd, Rs	0101	$r[Rd] - sw(f, Rs, d)$
MOV	Rd, Rs	0110	$r[Rd] = r[Rs]$
(reserved)		0111	/

SLL	Rd, d	1000	$r[Rd] = \text{shift_left_logical} (r[Rd], d)$
SLR	Rd, d	1001	$r[Rd] = \text{shift_left_rotate} (r[Rd], d)$
SRL	Rd, d	1010	$r[Rd] = \text{shift_right_logical} (r[Rd], d)$
SRA	Rd, d	1011	$r[Rd] = \text{shift_right_arithmetic} (r[Rd], d)$

4.1.2 入出力・制御命令

- IN (input) : 機器から入力した値をレジスタ Rd に格納する.
- OUT (output) : レジスタ Rs の値を機器に出力する.
- HLT (HLT) : SIMPLE を停止させる.

表 4.1.3 : SIMPLE の入出力・制御命令

mnemonic		op3	function
IN	Rd	1100	$r[Rd] = \text{input}$
OUT	Rs	1101	$\text{output} = (r[Rs], r[Rd], d)$
(reserved)		1110	/
HLT	/	1111	halt()

第2章で説明したように、d フィールドの値を指定することによって出力の場所が変わることができる。また、Rs と Rd で指定されたレジスタの値を同時に出力することができる。Rd が指定されていない時は 0000 を出力する。

4.2 ロード／ストア命令

15	14	13		11	10		8	7		4	3		0
op1			Ra			Rb			d				

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

- $I_{7:0}$ (d) 変位

SIMPLE のロード命令とストア命令の機能を表 2 に示す．ソース／デスティネーションは，フィールド **Ra** で指定されたレジスタ **Ra** である．また実行アドレスはベース・レジスタ・アドレス指定により，フィールド **Rb** で指定されたレジスタ **Rb** と，フィールド **d** を符号拡張した $\text{sign_ext}(d)$ を加算して求める．

- **LD (load)** : 主記憶アドレス ($r[\text{Rb}] + \text{sign_ext}(d)$) にある 16 ビットの値を **Ra** にロードする
- **ST (store)** : **Ra** にある 16 ビットの値を主記憶アドレス ($r[\text{Rb}] + \text{sign_ext}(d)$) のところに書き込む・

表 4.2 SIMPLE のロード・ストア命令

mnemonic		op1	function
LD	Ra, d(Rb)	00	$r[\text{Ra}] = *(r[\text{Rb}] + \text{sign_ext}(d))$
ST	Ra, d(Rb)	01	$*(r[\text{Rb}] + \text{sign_ext}(d)) = r[\text{Ra}]$

4.3 即値ロード／無条件分岐命令

15	14	13		11	10		8	7		4	3		0
10		op2			Rb			d					

- $I_{15:14}$ (10) 操作コード
 - $I_{13:11}$ (op2) 操作コード (000～110)
 - $I_{10:8}$ (Rb) ソース・デスティネーション・ベースのレジスタ番号
 - $I_{7:0}$ (d) 即値・変位
- **LI (load immediate)** : 即値 $\text{sign_ext}(d)$ をレジスタ **Rb** に格納する
 - **B (branch)** : **d** を符号拡張した値 $\text{sign_ext}(d)$ を変位として，PC 相対アドレス指定による分岐を行う

表 4.3 SIMPLE の即値ロード・無条件分岐命令

mnemonic		op1	function
LI	Rb, d	000	$r[Rb] = \text{sign_ext}(d)$
(reserved)		001	/
(reserved)		010	/
(reserved)		011	/
B	d	100	$PC = PC + 1 + \text{sign_ext}(d)$
(reserved)		101	/
(reserved)		110	/
(条件分岐命令)		111	表 4.4 参照

4.4 条件分岐命令形式

15	14	13		11	10		8	7		4	3		0
10				111			cond			d			

1. $I_{15:14}$ (10) 操作コード
2. $I_{13:11}$ (111) 操作コード
3. $I_{10:8}$ (cond) 分岐条件
4. $I_{7:0}$ (d) 変位

SIMPLE の条件分岐命令は以下の表に示すように、フィールド **cond** で定められる分岐条件が成り立てば PC 相対アドレスによる分岐を行い、成り立たなければ単に次の命令に移行する。各命令の分岐条件は以下の通り：

- BE (branch on equal to) : Z が 1
- BLT (branch on less than) : $S \wedge V$ が 1
- BLE (branch on less than or equal to) : Z または $(S \wedge V)$ が 1
- BNE (branch on not equal to) : Z が 0

表 4.4 SIMPLE の条件分岐命令

mnemonic		cond	function
BE	d	000	if (Z) PC = PC + 1 + sign_ext(d)
BLT	d	001	if (S ^ V) PC = PC + 1 + sign_ext(d)
BLE	d	010	if (Z (S ^ V)) PC = PC + 1 + sign_ext(d)
BNE	d	011	if (!Z) PC = PC + 1 + sign_ext(d)
(reserved)		100	/
(reserved)		101	/
(reserved)		110	/
(reserved)		110	/

4.5 ボードから直接入力される命令

reset (リセット信号)

- FPGA ボード上のプッシュスイッチを用いて、スイッチを押すと 1 が、離すと 0 が供給されるようにする。reset が 1 になると、PC 等をクリアし、初期状態に移行する。

exec (起動／停止信号)

- FPGA ボード上のプッシュスイッチを用いて、スイッチを押すと 1 が、離すと 0 が供給されるようにする。
- SIMPLE が停止状態にある時に exec が 0 から 1 に変化すると、SIMPLE は命令の実行を開始する。
- SIMPLE が実行状態にある時に exec が 0 から 1 に変化すると、その時点で実行中の命令が完了してから停止する。

第 5 章 構造と動作

この章では，さまざまな命令を受け取ったときの CPU の振る舞いについて説明する．同じような動作をする命令は，繰り返しを避けるためにまとめてある．第 5 章の説明については，図 5.1 と図 5.2 を参照してください．

図 5.1 : SIMPLE のブロック図

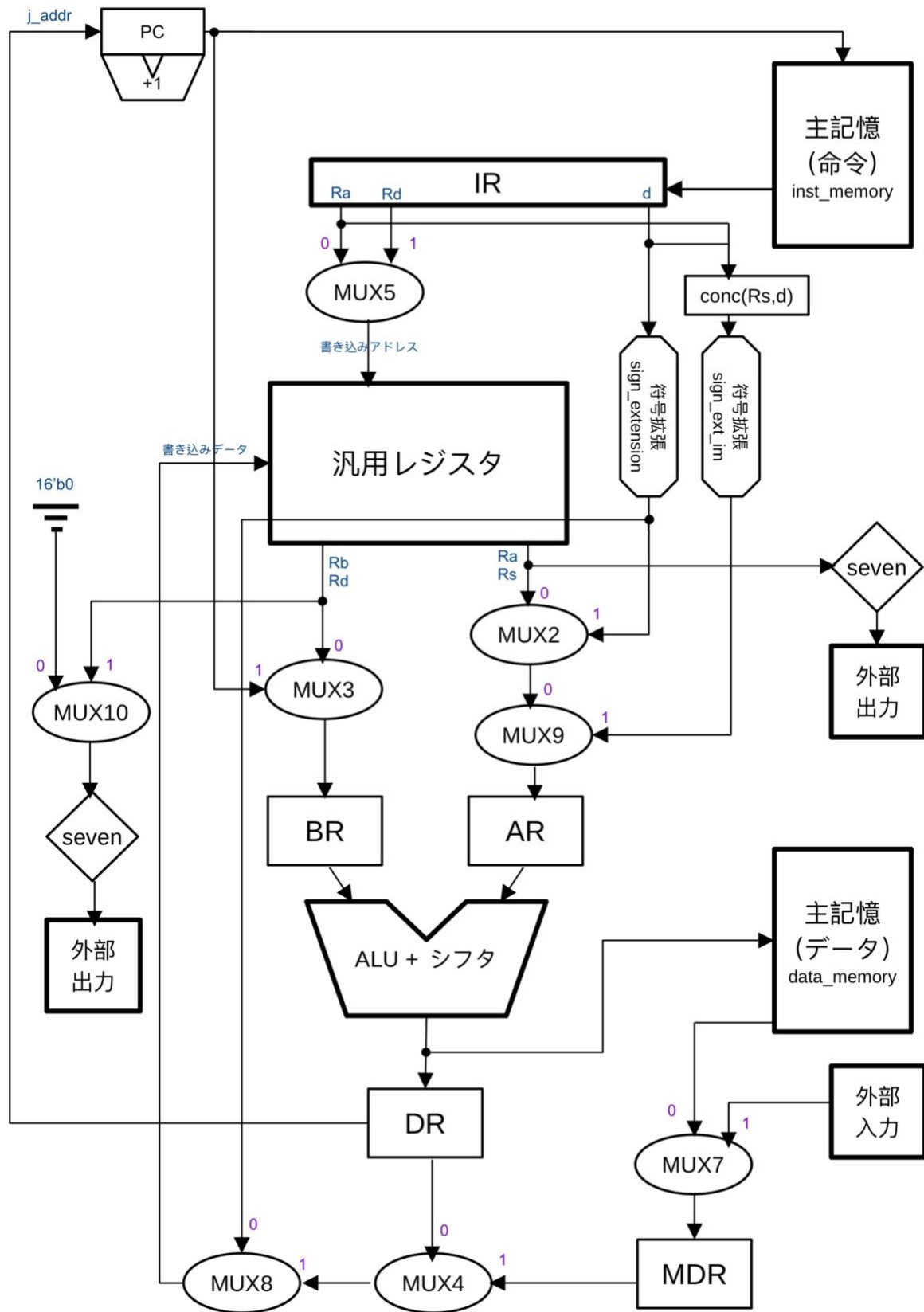
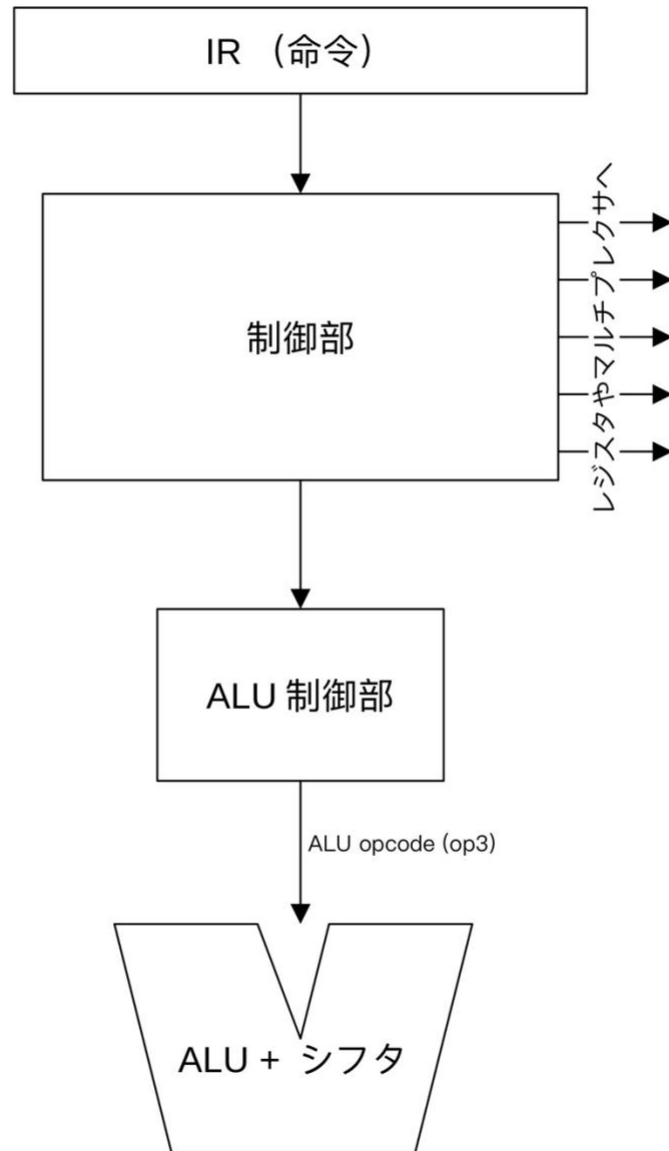


図 5.2 SIMPLE の制御ブロック図



ADD, SUB, AND, OR, XOR

1. PC (Program Counter) が保持するアドレスの命令を命令主記憶 inst_memory からフェッチし、IR (Instruction Register) に格納すると共に、PC に 1 を加える。IR にある命令が制御部に届き、制御部が ALU 制御部に必要な演算に対応するコードを送る。
 1. 命令の中のフラグ f が 0 ならば、マルチプレクサ MUX 3, MUX 2 と MUX 9 のセレクト信号として 0 が制御部から与えられ、IR が保持する命令の Rs フィールドと Rd フィールドで指定される汎用レジスタのアドレスにある値を読み出し、それぞれレジスタ AR と BR に格納する。
 2. フラグ f が 1 ならば、conc(Rs,d) (つまり Rs フィールドの 3 ビットと d フィールドの 3 ビットを結合した値) を符号拡張し、MUX 9 のセレクト信号が 1 にし、符号拡張した conc(Rs,d) が AR に格納する。一方、MUX 3 のセレクト信号が 0 で、Rd フィールドの値が BR に格納する。
2. AR と BR にある値が ALU の入力になる。ALU により、命令が定める演算を行い、その結果をレジスタ DR (Data Register) に格納する。条件コード S, Z, C, V をセットする。
3. DR にある値を命令の Rd フィールドに指定される汎用レジスタに書き込む。(MUX 4 のセレクト信号は 0 で、MUX 8 と MUX 5 のセレクト信号は 1 である。)

CMP 命令

1. PC が保持するアドレスの命令を主記憶からフェッチし、IR に格納すると共に、PC に 1 を加える。IR にある命令が制御部に届き、制御部が ALU 制御部に必要な演算に対応するコードを送る。
 1. 命令の中のフラグ f が 0 ならば、マルチプレクサ MUX 3, MUX 2 と MUX 9 のセレクト信号として 0 が制御部から与えられ、IR が保持する命令の Rs フィールドと Rd フィールドで指定される汎用レジスタのアドレスにある値を読み出し、それぞれレジスタ AR と BR に格納する。
 2. フラグ f が 1 ならば、conc(Rs,d) (つまり Rs フィールドの 3 ビットと d フィールドの 3 ビットを結合した値) を符号拡張し、MUX 9 のセレクト

信号が1にし、符号拡張した $\text{conc}(\text{Rs}, d)$ が AR に格納する。一方、MUX 3 のセクタ信号が0で、Rd フィールドの値が BR に格納する。

2. AR と BR にある値が ALU の入力になる。ALU により、命令が定める演算を行うが、結果を出力しない。条件コード S, Z, C, V をセットする。

MOV 命令

1. PC が保持するアドレスの命令を主記憶からフェッチし、IR に格納すると共に、PC に 1 を加える。IR にある命令が制御部に届き、制御部が ALU 制御部に必要な演算に対応するコードを送る。
2. マルチプレクサ MUX 3, MUX 2 と MUX 9 のセクタ信号として0が制御部から与えられ、IR が保持する命令の Rs フィールドと Rd フィールドで指定される汎用レジスタのアドレスにある値を読み出し、それぞれレジスタ AR と BR に格納する。
3. AR と BR にある値が ALU の入力になる。ALU により、命令が定める演算を行い、その結果をレジスタ DR に格納する。条件コード S, Z, C, V をセットする。
4. DR にある値を命令の Rd フィールドに指定される汎用レジスタに書き込む。
(MUX 4 のセクタ信号は0で、MUX 8 と MUX 5 のセクタ信号は1である。)

SLL, SLR, SRL, SRA 命令

1. PC が保持するアドレスの命令を主記憶からフェッチし、IR に格納すると共に、PC に 1 を加える。IR にある命令が制御部に届き、制御部が ALU 制御部に必要な演算に対応するコードを送る。
2. IR が保持する命令の、Rd フィールドで指定される汎用レジスタのアドレスにある値を読み出し、レジスタ BR に格納する (MUX 3 のセクタ信号は0である)。
3. MUX 2 のセクタ信号は1で MUX 9 のセクタ信号は0で、符号拡張した即値 d が AR に格納する。
4. AR と BR にある値が ALU の入力になる。ALU により、命令が定める演算を行い、その結果をレジスタ DR に格納する。条件コード S, Z, C, V をセットする。

5. DRにある値を命令の Rd フィールドに指定される汎用レジスタに書き込む。
(MUX 4 のセクタ信号は 0 で, MUX 8 と MUX 5 のセクタ信号は 1 である.)

IN 命令

1. PC が保持するアドレスの命令を主記憶からフェッチし, IR に格納すると共に, PC に 1 を加える. IR にある命令が制御部に届き, 制御部が制御信号を発信する.
2. 外部入力 (ディップスイッチ) の値が MDR に格納される. (ここで MUX 7 のセクタは 1 になる.)
3. MDR にある値を命令の Rd フィールドに指定される汎用レジスタに書き込む. (MUX 4 のセクタ信号は 1 で, MUX 8 と MUX 5 のセクタ信号は 1 である.)

OUT 命令

1. PC が保持するアドレスの命令を主記憶からフェッチし, IR に格納すると共に, PC に 1 を加える. IR にある命令が制御部に届き, 制御部が制御信号を発信する.
2. フィールド Rs と Rd で指定されたレジスタにある値が seven というデコーダの部品に通して, 7 SEG LED などの外部出力へ表示される. (ここで MUX10 のセクタは 1 になる.) Rd が指定されていない時は 4 桁の 0000 が表示される.

HLT 命令

1. PC が保持するアドレスの命令を主記憶からフェッチし, IR に格納すると共に, PC に 1 を加える.
2. IR にある命令が制御部に届き, 制御部が全ての部品を停止させる.

LD 命令

1. PC が保持するアドレスの命令を主記憶からフェッチし, IR に格納すると共に, PC に 1 を加える. IR にある命令が制御部に届き, 制御部が ALU 制御部に必要な演算に対応するコード (加算 ADD) を送る.

2. IR が保持する命令の、Rb フィールドで指定される汎用レジスタのアドレスにある値を読み出し、レジスタ BR に格納する。（ここで MUX 7 のセレクトは 0 になる。）
3. MUX 2 のセレクト信号は 1 で MUX 9 のセレクト信号が 0 で、符号拡張した即値 d が AR に格納する。
4. AR と BR にある値が ALU の入力になる。ALU により、加算を行い、その結果をレジスタ DR に格納する。
5. DR にある値をアドレスとして、PC に j_addr (jump address) として入力し、データ主記憶をアクセスする。読み出した値をレジスタ MDR (Memory Data Register) に格納する。（ここで MUX 7 のセレクトは 0 になる。）
6. MUX 4 と MUX 8 のセレクトが 1 になり、MDR にある値を命令の Ra フィールドに指定される汎用レジスタに書き込む。（MUX 5 のセレクトは 0 になる。）

ST 命令

1. PC が保持するアドレスの命令を主記憶からフェッチし、IR に格納すると共に、PC に 1 を加える。IR にある命令が制御部に届き、制御部が ALU 制御部に必要な演算に対応するコード（加算 ADD）を送る。
2. IR が保持する命令の、Rb フィールドで指定される汎用レジスタのアドレスにある値を読み出し、レジスタ BR に格納する。（ここで MUX 3 のセレクトは 0 になる。）
3. ここで MUX 2 のセレクトは 1 になり、MUX 9 のセレクトは 0 になり、符号拡張した即値 d が AR に格納する。
4. AR と BR にある値が ALU の入力になる。ALU により、加算を行い、その結果を直接にアドレスとしてデータ主記憶をアクセスする。
5. AR にある値がそのアドレスにストアする。

LI 命令

1. PC が保持するアドレスの命令を主記憶からフェッチし、IR に格納すると共に、PC に 1 を加える。
2. MUX 8 のセレクト信号が 0 になり、書き込みデータとして符号拡張した d を汎用レジスタに書き込む。書き込みアドレスを決めるために、MUX 5 のセレクト信号が 1 になる。

B, BE, BLT, BLE, BNE 命令

1. PC が保持するアドレスの命令を主記憶からフェッチし、IR に格納すると共に、PC に 1 を加える。IR にある命令が制御部に届き、制御部が ALU 制御部に必要な演算に対応するコード（加算 ADD）を送る。
2. MUX3 のセレクトは 1 で、PC+1 の値が BR に格納する。MUX 2 のセレクトは 1 で MUX 9 のセレクトは 0 になり、符号拡張した d の値が AR に格納する。
3. AR と BR にある値が ALU の入力になる。ALU のもう一つの入力になる。ALU が加算を行い、その結果をレジスタ DR に格納する。
4. DR にある値を分岐先アドレスとして PC に直接に書き込む。

reset（リセット信号）

1. 制御部へ 1 ビットの reset 信号が届く。
2. 制御部で SIMPLE を初期状態にさせる。

exec（起動／停止信号）

1. 制御部へ 1 ビットの exec 信号が届く。
2. 制御部でフラグを使って、その時点で実行中の命令が完了してから SIMPLE を停止する。