計算機科学実験及演習 3
9 班 CPU ユーザーズマニュアル

提出期限:2019/6/14
提出日: 2019 年 6 月 10 日

グループ番号: 9
1029293806　大山 偉永
1029286786　富村 勇貴

1　はじめに

この PDF は京都大学工学部情報学科計算機科学コース 2019 年度前期木曜 3,4,5 限、金曜 1,2,3,4,5 限の計算機科学実験 3 で我々 9 班が作成した CPU のユーザーズマニュアルである。約一ヶ月半での作成期間であり多くの拡張機能を持つわけではなく性能が高いということはできないが最低限の命令は実行することができる。以下その具体的なマニュアルである。

2　基本使用

完成日　2019/5/30
コア数　1
スレッド数　1
FMAX　74MHz
ワードサイズ　16bit
メモリサイズ　8,102B (4,096word × 2B)

3　マイクロアーキテクチャ

9 班 CPU はハーバードアーキテクチャとなっておりユーザーはデータメモリにデータを、命令メモリに命令を格納する必要がある。いづれも 1word:16bit で 4096word まで格納可能で命令列については次項を参照。また汎用レジスタは 8 つ備えており、r[0], r[1],···, r[7] と表記される。演算のソース／デスティネーションや主記憶のアドレス計算に用いられそのように命令のなかで使用する。アドレスについては変換などは行なわず、命令で定められるアドレスをそのまま主記憶のアドレスとする。またこの主記憶は、FPGA 上に搭載されている RAM にて構成する。また IN、OUT 命令については PowerMedusa MU500-RX/RK/7SEG 上の 8 ビット DIP スイッチ二個をつなげて 16bit の値を入力する。出力については LED:A0-A15 を用いて 16bit の値の出力を確認できる。また reset 命令はテンキースイッチの SW4 を割り当てた。ユーザーはこのスイッチを押すことで命令をはじめからやり直すことができる。また電源が入った時点でも自動的に reset されるようになっている。この reset ではプログラムカウンタの値は 0 になりレジスタの値も 0 に初期化されるがメモリの中身までは初期化できないのでユーザーはそこに注意する必要がある。さらに exec 信号には同様にテンキースイッチの SW5 を割り当てそれをおすことで命令を一時停止することができる。一度押すと命令の実行が一時停止されもう一度押すと再びそこから実行が再開される。

4　命令セットアーキテクチャ

9 班 CPU には 4 種類の命令形式があり各命令形式とフィールドの意味は以下の通りである。ユーザーは以下の定義に従って 16bit の命令列を命令メモリに格納することで命令が実行される。

4.1　演算／入出力命令

- $I_{15:14}(op1)$. . . . . 操作コード (11) (operation code, opcode)
- $I_{13:11}(Rs)$ . . . . . 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

9 班 CPU の演算／入出力命令を図 1 に示す。演算命令では結果に基づく条件コードが設定される。

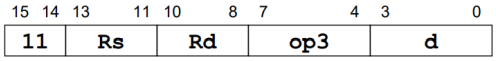
1. 算術演算
   レジスタ Rd と Rs の加算 (ADD: add) または減算 (SUB: subtract) の結果を Rd に格納し、条件コードを設定する。条件コード C には最上位ビットからの桁上げが設定される。
2. 論理演算
   レジスタ Rd と Rs の、ビットごとの論理積 (AND: and)、 論理和 (OR: or)、または排他的論理和 (XOR: exclusive-or) の結果を Rd に格納し条件コードを設定する。但し条件コード C は演算結果に関わらず 0 となる。
3. 比較演算 (CMP: compare)
   レジスタ Rd から Rs を減算し、結果に基づく条件コード設定のみを行なう。条件コード C には最上位ビットからの桁上げが設定される。
4. 移動演算 (MOV: move)
   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。但し条件コード C は Rs の値に関わらず 0 となる。
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。また条件コード C には、シフト桁数が 0 の時または SLR では 0 が、それ以外では最後にシフト・アウトされたビットの値が設定される。条件コード V は常に 0 が設定される。

6. 入出力命令
   - IN (input) . . . . . スイッチなどの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 7SEG LED などの機器に出力する。
   - HLT (halt) . . . . . SIMPLE を停止させる。

   なお “(reserved)” と記された命令は何の動作もせず、単に次の命令に移行する。



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

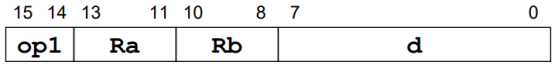
図 1　9 班 CPU の演算／入出力命令

4.2　ロード／ストア命令

- $I_{15:14}(op1)$ . . . . . 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 (displacement)

9 班 CPU のロード命令 (LD: load) とストア命令 (ST: store) の機能を図 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。また実効アドレスはベー

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 sign ext(d) を加算して求める。



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

図 2　9 班 CPU のロード／ストア命令

4.3　即値ロード／即値演算／無条件分岐命令／ジャンプ命令

- $I_{15:14}(op1)$ . . . . . 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／ベースのレジスタ番号
- $I_{7:0}(d)$. . . . . 即値または変位

9 班 CPU の即値ロード命令 (LI: load immediate) 、即値加算命令 (ADDI: add immediate)、ジャンプ命令 (JMP: jump)、無条件分岐命令 (B: branch)、即値比較命令（CMPI:compare immediate）の機能を図 3 に示す。

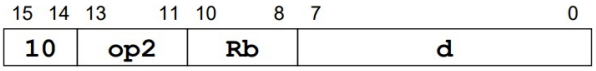
- LI . . . . . 即値 sign ext(d) をレジスタ Rb に格納する。
- ADDI . . . . 即値 sign ext(d) とレジスタ Rb の値を加算しその結果を Rd に格納し条件コードを設定する。
- JMP . . . . . 符号拡張した d の値をアドレスとして PC のアドレス指定による分岐を行う。
- B . . . . . d を符号拡張した値を変位として、PC 相対アドレス指定による分岐を行なう。
- CMPI . . . . . 即値 sign ext(d) とレジスタ Rb の値を比較しその結果を Rd に格納し条件コードを設定する。

010 addi 011-¿jump 101 cmpi

4.4　条件分岐命令／ nop 命令

- $I_{15:14}(op1)$ . . . . . 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

9 班 CPU の条件分岐命令は図 4 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

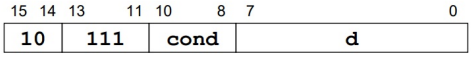


mnemonic	op2	function
LI Rb,d	000	r[Rb] = sign_ext(d)
(reserved)	001	
ADDI	010	r[Rb] = sign_ext(d) + R[Rb]
JMP	011	PC = sign_ext(d)
B d	100	PC = PC + 1 + sign_ext(d)
CMPI	101	r[Rb] = r[Rb] - sign_ext(d)
(reserved)	110	
(条件分岐命令)	111	表 4 参照

図 3　9 班 CPU の即値ロード／即値演算／無条件分岐命令／ジャンプ命令

てば PC 相対アドレスによる分岐、または nop 命令（何の動作もしない命令）を行ない、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) . . . . . 条件コード Z が 1
- BLT (branch on less-than) . . . . . 条件コード S と V の XOR(S^ V) が 1
- BLE (branch on less-than or equal-to) . . . Z または (S^ V) が 1
- BNE (branch on not-equal-to) . . . . . 条件コード Z が 0



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)
NOP	100	(no operation)
(reserved)	101	
(reserved)	110	
(reserved)	111	

図 4　9 班 CPU の条件分岐命令