最終成果物のユーザーズマニュアル

グループ20 藤田陽斗

2021年6月3日

第1章　概要

team20 は simple アーキテクチャをもとにした 5 段パイプライン処理プロセッサを設計した。プロセッサの基本的な仕様は下表 1.1 の通り。

ビット数	16 bit / word
コア数	1
パイプラインサイクル	80 MHz
データメモリ	4096 words
命令メモリ	4096 words

表 1.1: プロセッサの基本的な仕様

以下、この章ではプロセッサの入出力やメモリ等、使用する上でのインターフェースになる部分の仕様について説明する。

1.1　入出力の仕様

下表 1.2 はプロセッサの各入出力の仕様を示している。

表 1.2: プロセッサの入出力の仕様

入出力	仕様
clock	プロセッサを制御するクロック信号. 内部で PLL 回路を用いて発振し、4 倍周波数のクロック信号を生成して動作させるため、20 MHz のクロック信号の入力が推奨される。
reset	リセット信号．reset が 1 のとき，すべてのレジスタは 0 に初期化され，reset が 0 に戻ると命令メモリの 0 番地からリセット後の処理が開始される. 入力機器としてディップスイッチを割り当てることが推奨される
out	外部出力信号. OUT 命令により指定されたレジスタの値が出力される。

1.2　メモリの仕様

下表 1.3 はプロセッサの各メモリの仕様を示している。

表 1.3: プロセッサのメモリの仕様

メモリ	仕様
命令メモリ	アドレス空間は 4096 語で語単位にアドレスがつけられている。プロセッサはこのメモリの 0 番地に書かれた命令から順に実行する。inst.mif ファイルに命令を書き込むことで、命令メモリを構成できる。
データメモリ	アドレス空間は 4096 語で語単位にアドレスがつけられている。

1.3　レジスタの仕様

下表 1.4 はプロセッサの各レジスタの仕様を示している。

表 1.4: プロセッサのメモリの仕様

レジスタ	仕様
汎用レジスタ	7 個の汎用レジスタが備えられ、 r[1], r[2], ..., r[7] と表記される。命令のオペランドとして指定される。
ゼロレジスタ	常に値が 0 のレジスタ。r[0] と表記され、命令のオペランドとして指定される。また、書き込み先としてこのレジスタを指定した場合、その値は捨てられる。
プログラムカウンタ	実行中の命令のアドレスを保持する．PC と表記する．
条件コード	演算命令の結果に基づく分岐条件を保持する 4 個のレジスタ S(sign), Zero(zero), C(carry), V(overflow) からなり，下表 1.5 のように設定される．

表 1.5: 条件コードの仕様

条件コード	仕様
S	演算結果が負ならば 1，そうでなければ 0
Zero	演算結果がゼロならば 1，そうでなければ 0
C	演算結果に桁上げがあれば 1，そうでなければ 0
V	演算結果が符号付き 16 ビットで表せる範囲を越えた場合 1，そうでなければ 0

第2章　プロセッサの特長

設計したプロセッサの特長を以下に挙げる。

- このプロセッサは simple アーキテクチャを拡張した設計になっており、simple プロセッサで実行可能な、算術/シフト演算命令、分岐命令、停止命令、ロード/ストア命令、即値ロード命令、入出力命令等のすべての命令を実装している。
- 即値加算/比較命令を実装しており、ソートを始めとしたループ処理の多いプログラムを高速に実行できる。
- ジャンプ＆リンク命令を実装しており、手続き呼び出しを含むプログラムを実行できる。
- パイプライン化を簡単にするためにハーバードアーキテクチャを採用しており、プログラム内蔵方式のものと比べると、メモリの柔軟性は劣る。
- 5 段パイプライン処理を実現しておりプログラムを高速に実行できる。
- パイプライン処理に伴う制御ハザード、データハザードに適切に対応することができ、ユーザは命令系列の実行順を気にせずにプログラムを書くことができる。
- ゼロレジスタを設けており、ジャンプ＆リンク命令と組み合わせて無条件分岐を実現できる。

第3章　命令セットアーキテクチャ

以下に設計したプロセッサの命令セットアーキテクチャを示す。

3.1　命令形式

命令のビット数はすべて 16 ビットの固定長であり、以下に示すような 4 つの命令形式がある。以下、命令 i ビット目から j ビット目までのフィールドを I[i:j] と表す。

3.1.1　命令形式 1

- 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

3.1.2　命令形式 2

- 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 (signed)

3.1.3 命令形式 3

- I[15:14] (op1) : 操作コード (10)
- I[13:11] (op2) : 操作コード (000 ≤ op2 ≤ 110)
- I[10:8] (Rb) : ソース/デスティネーション/デスティネーションレジスタ番号
- I[7:0] (d) : 即値または変位 (signed)

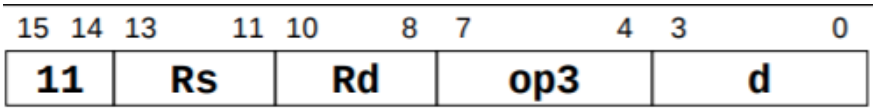
3.1.4 命令形式 4

- I[15:14] (op1) : 操作コード (10)
- I[13:11] (op2) : 操作コード (111)
- I[10:8] (cond) : 分岐命令
- I[7:0] (d) : 変位 (signed)

以下では各命令形式について、詳細を示す。

3.2　命令形式 1

各命令と機械語の対応は以下の通り。ただし、reserved とある機械語は何の命令にも対応していない。



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)
(reserved)	1100	
(reserved)	1101	
JALR	1110	r[Rd] = PC + 1, PC = r[Rs]
HLT	1111	halt()

表 3.1: 命令形式 1 に従う命令の命令セットアーキテクチャ

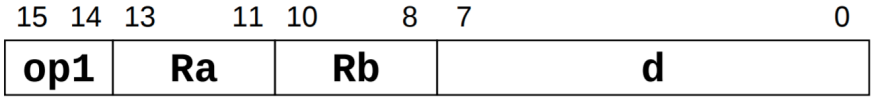
各命令の仕様は下表 3.2 の通り。

表 3.2: シフト演算命令の仕様

命令（演算）	仕様
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．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．
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．但し条件コード C は演算結果に関わらず 0 となる．
比較演算	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．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．
移動演算	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．
シフト演算	レジスタ Rd の値を，下表**??**のように即値 d だけシフトした値を Rd に格納し，条件コードを設定する. シフト桁数は即値 d ($0 \leq d \leq 15$) 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．
ジャンプ＆リンク命令	レジスタをロードするジャンプ＆リンク命令。PC+1 の値を Rd フィールドで指定されたレジスタに退避し、PC に Rs フィールドで指定されたレジスタの値をロードする。

3.3 命令形式 2

各命令と機械語の対応は以下の通り。



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

表 3.3: 命令形式 2 に従う命令の命令セットアーキテクチャ

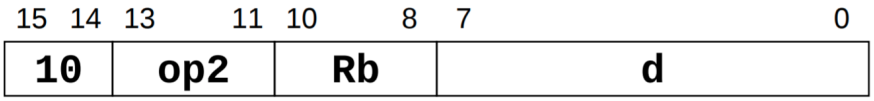
各命令の仕様は下表 3.4 の通り。

表 3.4: 命令形式 2 に従う命令の仕様

命令	仕様
LD	レジスタ Rb と 即値 d を符号拡張した値を加算して求めたアドレスから、レジスタ Ra に値をロードする.
ST	レジスタ Rb と 即値 d を符号拡張した値を加算して求めたアドレスに、レジスタ Ra の値をストアする.

3.4 命令形式 3

各命令と機械語の対応は以下の通り。



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

表 3.5: 命令形式 3 に従う命令の命令セットアーキテクチャ

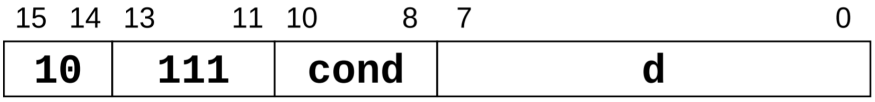
各命令の仕様は下表 3.6 の通り。

表 3.6: 命令形式 3 に従う命令の仕様

命令	仕様
LI	即値 d を符号拡張した値をレジスタ Rb に格納する.
ADDI	即値加算命令。8 ビットの即値フィールドに与えられた値を 16 ビットに拡張して指定されたレジスタの値に加える。
CMPI	即値比較命令。8 ビットの即値フィールドに与えられた値を 16 ビットに拡張して指定されたレジスタの値と比較し、結果に応じて条件コードをセットする。
JAL	ジャンプ&リンク命令。PC+1 の値を指定されたレジスタに退避し、8 ビットの即値フィールドに与えられた値を 16 ビットに拡張して PC の値に加える。

3.5 命令形式 4

各命令と機械語の対応は以下の通り。



mnemonic	cond	function
BE d	000	if (Z) PC = PC + sign_ext(d)
BLT d	001	if (S ^V) PC = PC + sign_ext(d)
BLE d	010	if (Z || (S ^ V)) PC = PC + sign_ext(d)
BNE d	011	if (!Z) PC = PC + sign_ext(d)
(reserved)	100	
(reserved)	101	
(reserved)	110	
(reserved)	111	

表 3.7: 命令形式 4 に従う命令の命令セットアーキテクチャ

各命令の仕様は下表 3.8 の通り。

表 3.8: 命令形式 4 に従う命令の仕様

命令	仕様
BE	条件コード Zero が 1 ならば、PC 相対アドレシングによる分岐を行う。
BLT	条件コード S と V の排他的論理和が 1 ならば、PC 相対アドレシングによる分岐を行う。
BLE	条件コード Zero か S と V の排他的論理和が 1 ならば、PC 相対アドレシングによる分岐を行う。
BNE	条件コード Zero が 0 ならば、PC 相対アドレシングによる分岐を行う。