LWZeta 命令セットアーキテクチャ仕様(Ver0.94β版)

平山 直紀

https://wzeta.idletime.tokyo/

1. 概要

WZeta は超軽量な 8bit CPU です。大した CPU ではありません。その劣化版が LWZeta (Lesser WZeta)です。アセンブラレベルで完全互換です。WZeta の特長が他人の特許を侵害していないことを確認するまで LWZeta を使ってプログラム開発をすることが可能です。WZeta ではプログラムの転送に 8bit の配線でしたが LWZeta は 32bit の配線となっていて WZeta のメリットの部分が削除されています。これで WZeta のアイディアが回避できている根拠について。32bit の命令コードのプログラムをバイト単位でメモリに格納しておきます。 このとき命令コードのフォーマットを工夫しておけば、 プログラムカウンタによって順番に CPU に到着して、到着した部分だけで実行ができる。 工夫がない場合、最悪 32bit すべての命令コードが CPU に到着しなければ命令を実行できない。 劣化版では命令コードのフォーマットをシャッフルすることで工夫がない命令コードと判断できる。

2. 基本仕様

・命令コード 1 ワード 32bit 固定長
・すべての命令コードは 4 サイクルで実行される。
・汎用レジスタ　128 本
・プログラムメモリ　512 ワード(最大 16K ワード)
・データメモリ 2KB(最大リード 64KB、ライト 32KB)
　128 バイト目以降のアクセスにはオプションの FETCH 命令と STORE 命令でアクセス
・サブルーチンコール　1 段（最大∞）
・8bit の I/O ポートと 8bit のバス
・2 の補数

3. レジスタ、フラグ

#	名前	bit 幅	用途
1	R0-R127	8	汎用レジスタ
2	CF	1	キャリーフラグ
3	ZF	1	キャリーフラグ
4	PC	11-16	プログラムカウンタ(ワードアドレスではなくバイト)

4. 命令コードフォーマット

32bit 固定長の命令コード。

bit31	bit30	bit29	bit28	bit27	bit26	bit25	bit24
ストア	ライトする汎用レジスタ番号						

bit23	bit22	bit21	bit20	bit19	bit18	bit17	bit16
B 選択	即値 n / リードする汎用レジスタ番号						

bit15	bit14	bit13	bit12	bit11	bit10	bit9	bit8
A 選択	即値 n / リードする汎用レジスタ番号						

bit7	bit6	bit5	bit4	bit3	bit2	bit1	bit0
演算選択	オペコード						

CPU の内部に A レジスタと B レジスタがある。すべての命令は A レジスタと B レジスタに値を代入する。命令によっては代入した値を使わないものもあるが大半の命令は使用する。分岐命令も、分岐先のアドレスを A、B レジスタに代入する。

CPU 基本動作は A、B レジスタに値を代入して A と B を入力とする演算を行い結果を汎用レジスタに書き込む。A レジスタへの代入は即値 n(0～127)の値か、汎用レジスタ 0-127 番の値が代入できる。どちらを代入するかは bit7 で決まる。B レジスタも同様で bit15 で即値か汎用レジスタを選択。

即値は 0～127 までだが、加算命令で n + n + 1 ができるので 0～255 の即値を 1 命令で汎用レジスタに書き込める。

1 命令を 4 サイクル実行する基本的な流れ

最上位 bit	ステージ 0	ステージ 1	ステージ 2	ステージ 3
0	A=n	B=n	ans = A ∘ B	-
1	A=R[n]	B=R[n]	ans = IN ∘ B	R[n] = ans

ステージ 2 の IN は 8bit の I/O 入力を取得。

分岐命令

最上位 bit	ステージ 0	ステージ 1	ステージ 2	ステージ 3
0	A=n	B=n	nn に分岐	空白
1	A=R[n]		(空)	

分岐アドレスの下位 7bit には汎用レジスタの値を使うことができるが上位 7bit は即値。

5. オペコード

ステージ 2 の bit6-0、7bit がオペコード。一般のオペコードと異なりデコードすることなく極力面積が小さくなるような形式になっています。

bit6	bit5	bit4	bit3
加算/論理の選択	RETURN	JUMP	CALL(bit4 と併用)

分岐条件

bit2	bit1	bit0
0: 無条件 1: CF 2: ZF		CF/ZF の条件

加算系

bit2	bit1	bit0
CF	NOT	ONE

論理系

bit2	bit1	bit0
0	0: A 1: AND 2: XOR 3: OR	
1	0: STORE 1: FETCH (オプション)	

I/O 系

bit2	bit1	bit0
3'b111 の場合、OUTPUT 命令 A レジスタはポート ID、B レジスタの値を I/O に出力する		

6. ニモニック

x,y は 7bit の即値、n は 8bit の即値

ニモニック	意味
NOP	何もしない
LOAD Rz , Rx	レジスタ z にレジスタ x の値を代入
LOAD Rz, n	レジスタ z に即値 n を代入
ADD Rz , Rx , Ry	Rz = Rx + Ry
ADD Rz , x , Ry	Rz = x + Ry
ADD Rz , Rx , y	Rz = Rx + y
ADD Rz , x , y	Rz = x + y
ADDC Rz , Rx , Ry	Rz = Rx + Ry + CF
ADDC Rz , x , Ry	Rz = x + Ry + CF
ADDC Rz , Rx , y	Rz = Rx + y + CF
ADDC Rz , x , y	Rz = x + y + CF
SUB Rz , Rx , Ry	Rz = Rx – Ry
SUB Rz , x , Ry	Rz = x – Ry
SUB Rz , Rx , y	Rz = Rx - y
SUB Rz , x , y	Rz = x - y
SUBC Rz , Rx , Ry	Rz = Rx - Ry with CF
SUBC Rz , x , Ry	Rz = x - Ry with CF
SUBC Rz , Rx , y	Rz = Rx - y with CF
SUBC Rz , x , y	Rz = x - y with CF
AND Rz , Rx , Ry	Rz = Rx & Ry
AND Rz , x , Ry	Rz = x & Ry
AND Rz , Rx , y	Rz = Rx & y
AND Rz , x , y	Rz = x & y
OR Rz , Rx , Ry	Rz = Rx | Ry
OR Rz , x , Ry	Rz = x | Ry
OR Rz , Rx , y	Rz = Rx | y
OR Rz , x , y	Rz = x | y
XOR Rz , Rx , Ry	Rz = Rx ^ Ry
XOR Rz , x , Ry	Rz = x ^ Ry
XOR Rz , Rx , y	Rz = Rx ^ y
XOR Rz , x , y	Rz = x ^ y

TEST Rx , Ry	Rx & Ry → ZF
TEST x , Ry	x & Ry → ZF
TEST Rx , y	Rx & y → ZF
TEST x , y	x & y → ZF
COMPARE Rx , Ry	Rx - Ry → CF , ZF
COMPARE x , Ry	x - Ry → CF , ZF
COMPARE Rx , y	Rx - y → CF , ZF
COMPARE x , y	x - y → CF , ZF
COMPAREC Rx , Ry	Rx - Ry → CF , ZF
COMPAREC x , Ry	x - Ry → CF , ZF
COMPAREC Rx , y	Rx - y → CF , ZF
COMPAREC x , y	x - y → CF , ZF
JUMP yx	yx に無条件分岐
JUMP y , Rx	yRx に無条件分岐、Rx は下位 7bit
JUMPC0 yx	CF=0 のとき yx に分岐
JUMPC0 y , Rx	CF=0 yRx に分岐、Rx は下位 7bit
JUMPC1 yx	CF=1 のとき yx に分岐岐
JUMPC1 y , Rx	CF=1 yRx に分岐、Rx は下位 7bit
JUMPNZ yx	ZF=0 のとき yx に分岐
JUMPNZ y , Rx	ZF=0 yRx に分岐 , Rx は下位 7bit
JUMPZ yx	ZF=1 のとき yx に分岐
JUMPZ y , Rx	ZF=1 yRx に分岐 , Rx は下位 7bit
CALL yx	yx にコール
CALL y , Rx	yRx にコール 、Rx は下位 7bit
CALLC0 yx	CF=0 のとき yx にコール
CALLC0 y , Rx	CF=0 yRx にコール , Rx は下位 7bit
CALLC1 yx	CF=1 のとき yx にコール
CALLC1 y , Rx	CF=1 yRx にコール , Rx は下位 7bit
CALLNZ yx	ZF=0 のとき yx にコール
CALLNZ y , Rx	ZF=0 yRx にコール , Rx は下位 7bit
CALLZ yx	ZF=1 のとき yx にコール
CALLZ y , Rx	ZF=1 yRx にコール , Rx は下位 7bit
RETURN	リターン
RETURNC0	CF=0 のときリターン
RETURNC1	CF=1 のときリターン

RETURNNZ	ZF=0 のときリターン
RETURNZ	ZF=1 のときリターン
LOAD＆RETURN Rz , N	z に即値 N を代入してリターン
INPUT Rz , Rx	ポート ID Rx の I/O 入力を Rz に代入
INPUT Rz , x	ポート ID x の I/O 入力を Rz に代入
INPUTAND Rz , Rx , Ry	ID Rx の I/O 入力と Ry を AND して Rz に代入
INPUTAND Rz , Rx , y	ID Rx の I/O 入力と y を AND して Rz に代入
INPUTAND Rz , x , Ry	ID x の I/O 入力と Ry を AND して Rz に代入
INPUTAND Rz , x , y	ID x の I/O 入力と y を AND して Rz に代入
INPUTOR Rz , Rx , Ry	ID Rx の I/O 入力と Ry を OR して Rz に代入
INPUTOR Rz , Rx , y	ID Rx の I/O 入力と y を OR して Rz に代入
INPUTOR Rz , x , Ry	ID x の I/O 入力と Ry を OR して Rz に代入
INPUTOR Rz , x , y	ID x の I/O 入力と y を OR して Rz に代入
INPUTXOR Rz , Rx , Ry	ID Rx の I/O 入力と Ry を XOR して Rz に代入
INPUTXOR Rz , Rx , y	ID Rx の I/O 入力と y を XOR して Rz に代入
INPUTXOR Rz , x , Ry	ID x の I/O 入力と Ry を XOR して Rz に代入
INPUTXOR Rz , x , y	ID x の I/O 入力と y を XOR して Rz に代入
INPUTTEST Rx , Ry	ID Rx の I/O 入力と Ry を AND して ZF に反映
INPUTTEST Rx , y	ID Rx の I/O 入力と y を AND して ZF に反映
INPUTTEST x , Ry	ID x の I/O 入力と Ry を AND して ZF に反映
INPUTTEST x , y	ID x の I/O 入力と y を AND して ZF に反映
OUTPUT Rx , Ry	ポート ID Rx、データ Ry を I/O に出力
OUTPUT x , Ry	ポート ID x 、データ Ry を I/O に出力
OUTPUT x , y	ポート ID x 、データ y を I/O に出力

7. シフト命令の代替法

ニモニック	動作
ADD Rz , Ra , Ra	Rz = Ra << 1
ADD Ra , Ra , Ra	Ra = Ra << 1

8. キャリーフラグ(CF)とゼロフラグ(ZF)

ADD,ADDC,SUB,SUBC,COMPARE,COMPAREC 命令は演算結果により CF,ZF が設定されます。AND,OR,XOR 命令は演算結果によって ZF が設定されます。CF,ZF は次の命令で

のみ利用可能です。(全命令で CF,ZF がセットされますが意味があるのは前述の演算命令のみ)

9. SUB によるキャリーフラグ(CF)の変化

CPU によっては引き算した結果が負になる場合に CF=1 をセットするものもありますが WZeta では引き算した結果が負になる場合は CF=0、正または 0 の場合は CF=1 がセットされます。

10. 比較

COMPARE 命令を使って比較をする。(第一オペランド) ≧ (第二オペランド)の場合、CF=1 になる。2 バイト以上の数の比較は初回、COMPARE 命令、2 回目以降、COMPAREC 命令を使います。

11. FETCH 命令(オプション)

命令コードの 1 バイト目では A レジスタに値を代入するためデータメモリにアクセスに行きます。1 バイト目の最上位ビットは即値かレジスタの読み出しかを選択するビットで使用されているためデータメモリの 128 バイト目までしかアクセスできません。A レジスタへ読み出すアドレスを 16bit(B レジ+A レジ)に変更する命令が FETCH 命令です。16bit アドレスで読み出したい命令の前に FETCH 命令を置けば、A レジスタに FETCH 命令で設定した A レジ、B レジの値をアドレスとした値が代入されます。ニモニックでいうと R(x)が R(B+A)になります。

x,y は 7bit の即値

ニモニック	動作
FETCH Rx , Ry	R(RyRx)のフェッチ
FETCH , x , Ry	R(Ry x)のフェッチ
FETCH Rx , y	R(yRx)のフェッチ
FETCH x , y	R(yx)のフェッチ

x,y は 7bit のため即値ではアクセスできない領域があることに注意

分岐系の命令では yx は 14bit を表していますが、()内は 16bit

12. STORE 命令(オプション)

通常の命令は命令コードの 4 バイト目の bit6-0 をアドレスとしてデータメモリに演算結果を書き込みます。STORE 命令は 15bit(4 バイト目の bit6-0 + B レジスタ)のアドレスに A レジスタの値を書き込みます。

x は 7bit の即値

ニモニック	動作
STORE Rz , Rx , Ry	R(zRy) = Rx
STORE Rz , x , Ry	R(zRy) = x
STORE Rz , Rx , y	R(zy) = Rx
STORE Rz , x , y	R(zy) = x

分岐系の命令では xy は 14bit を表していますが、()内は 16bit

13. OUTPUT2 命令(オプション)

OUTPUT 命令で 8bit のポート ID が不足したとき、この命令を使います。

オペコード（bit6, bit2, bit1, bit0) = (1, 1, 1, 0)