powerutil Documentation

Book Moons

Jul 12, 2018

Contents

1 Loading **3**
- 1.1 Initialization Parameters . . . 3

2 Extended Directives **5**
- 2.1 Error . . . 5
- 2.2 Parameter Validation . . . 6
- 2.3 Symbol . . . 6

3 Macros **7**
- 3.1 Function . . . 7
- 3.2 Load . . . 8
- 3.3 Logic . . . 10
- 3.4 Math . . . 11

4 Symbols **17**
- 4.1 File Descriptor . . . 17
- 4.2 Register . . . 17

Power Architecture assembly utilities.
Assembly library for writing Power code.

Source: https://gitlab.com/powerutil/powerutil
Bug reports: https://gitlab.com/powerutil/powerutil/issues

Features

- Define slim functions
- Unaligned vector load
- Symbolic memory load
- Convenience math and logic operations
- Register name symbols
- Validate macro arguments

Usage

```
.include "powerutil.s"
powerutil endian=big

/* powerutil API available, with big endian storage access */
```

Contents

CHAPTER 1

Loading

powerutil is a library of assembly code. The library is provided as a single file for including into your code. An initialization macro takes parameters through which you specify things like byte endianness for storage access.

To use *powerutil* include the `powerutil.s` file and call the `powerutil` macro with your desired configuration.

```
.include "powerutil.s"
powerutil endian=little

/* powerutil API available, with little endian storage access */
```

1.1 Initialization Parameters

`powerutil endian?`

Initialize *powerutil* with specified configuration.

Parameters **`endian`** (*`string`*) – Byte endianness for storage access. Allowed values "big" or "little". Optional. No default. Leaving unspecified disables the endianness abstraction layer (abstracted macros like `lvxu` will be unavailable). Explicit endianness macros are always available (for example, `lvxu_be` or `lvxu_le`).

CHAPTER 2

Extended Directives

2.1 Error

2.1.1 ..error - Error

..error message

Emit assembly error.
Message is required. This eliminates opaque errors with no message.
Preferred over the standard error directives.

Type assembly.

Parameters **message** (*string*) – Error message. Required. Input only.

Raises **Error** – Always. With provided message.

2.1.2 ..errord - Error with default

..errord message? default

Emit assembly error.
Use error message if provided, default error message otherwise.
For use by macros that take an optional error message.

Type assembly.

Parameters

- **message** (*string*) – Error message. Optional. No default. Input only.
- **default** (*string*) – Default error message. Required. Input only.

Raises **Error** – Always. With provided message or default.

2.2 Parameter Validation

Parameter validation directives raise an error if validation fails.

2.2.1 ..noteq - Validate not equal numeric

```
..noteq first second message?
```

Verify 2 values not equal numerically.

Type assembly.

Parameters

- **first** (*number*) – First value. Required. Input only.
- **second** (*number*) – Second value. Required. Input only.
- **message** (*string*) – Error message. Optional. No default. Input only.

Raises **Error** – If first is equal to second.

2.3 Symbol

2.3.1 ..def - Define internal symbol

```
..def name value
```

Define symbol for internal use.
Make symbol local, hidden, and permanent. Redefinitions not possible.

Type assembly.

Parameters

- **name** (*symbol*) – Symbol name. Receives value. Required. Output only.
- **value** (*AnyAssembly*) – Symbol value. Required. Input only.

CHAPTER 3

Macros

3.1 Function

3.1.1 entry_minimal - Minimal function entry

entry_minimal name

Emit minimal function entry sequence.
No stack frame. No TOC pointer.
Minimizes memory access. Minimizes computation.

Type transcendent.
Changes section to `.text`.
No implicit register usage.

Parameters **name** (*string*) – Function name. Required. Input only.

3.1.2 entry_simple - Simple function entry

entry_simple name

Emit simple function entry sequence.
No stack frame. TOC pointer initialized.
Minimizes memory access.

Type transcendent.
Changes section to `.text`.

Register effects
r2 TOC pointer

Parameters **name** (*string*) – Function name. Required. Input only.

3.1.3 exit_minimal - Minimal function exit

exit_minimal name

Close function opened with `entry_minimal`.
No stack frame destruction.

Type code.
No implicit register usage.

Parameters **name** (*string*) – Function name. Must match name given in the paired `entry_minimal`. Required. Input only.

3.1.4 exit_simple - Simple function exit

exit_simple name

Close function opened with `entry_simple`.
No stack frame destruction.

Type code.
No implicit register usage.

Parameters **name** (*string*) – Function name. Must match name given in the paired `entry_simple`. Required. Input only.

3.2 Load

3.2.1 lma - Load symbol address

lma rt sym

Load symbol address into general register.
Summary: rt = &sym

Type code.
No implicit register usage.

Parameters

- **rt** (*GeneralRegister*) – Receives address. Required. Output only.
- **sym** (*symbol*) – Symbol to load address of. Required. Input only.

3.2.2 lmbz - Load symbol byte and zero

lmbz rt sym d?

Load zero extended byte from symbol into general register.
Summary: rt = ext0(byte sym.d)

Type code.
No implicit register usage.

Parameters

- **rt** (*GeneralRegister*) – Receives value. Required. Output only.
- **sym** (*symbol*) – Symbol to load from. Variable address addend. Required. Input only.
- **d** (*SignedInteger16b*) – Displacement. Byte offset from symbol. Constant address addend. Optional. Default 0.

3.2.3 lvxu - Load vector indexed unaligned

lvxu vt ra rb ri? vi? vp?

Load vector register from potentially unaligned memory.
Indexed address specification.
Works for aligned and unaligned memory.

Type code.
Abstracts on `endian`.
Leaves undefined: ri vi vp

Register usage
ri Address calculation
vi Intermediate load
vp Permute control

Parameters

- **vt** (*VectorRegister*) – Receives value. Required. Output only.
- **ra** (*0|GeneralRegister*) – Address addend. Required. Input only.
- **rb** (*GeneralRegister*) – Address addend. Required. Input only. May not be r0.

- **ri** (*GeneralRegister*) – Address calculation. May not be ra. Optional. Default r11. Internal.
- **vi** (*VectorRegister*) – Intermediate load. May not be vt. Optional. Default v17. Internal.
- **vp** (*VectorRegister*) – Permute control. May not be vt vi. Optional. Default vp. Internal.

Raises

- **Error** – If ri is ra. Usage conflicts.
- **Error** – If vi is vt. Usage conflicts.
- **Error** – If vp is vt. Usage conflicts.
- **Error** – If vp is vi. Usage conflicts.

3.2.4 lvxu_be - Load vector indexed unaligned, big endian

lvxu_be vt ra rb ri? vi? vp?

Big endian variant of `lvxu`.

3.2.5 lvxu_le - Load vector indexed unaligned, little endian

lvxu_le vt ra rb ri? vi? vp?

Little endian variant of `lvxu`.

3.3 Logic

3.3.1 clr - Clear

clr rt

Clear general register to 0.
Summary: rt = 0

Type code.
No implicit register effects.

Parameters **rt** (*GeneralRegister*) – Register to clear. Required. Output only.

3.3.2 vclr - Vector clear

vclr vt

Clear vector register to 0.
Summary: vt = 0

Type code.
No implicit register effects.

Parameters **vt** (*VectorRegister*) – Register to clear. Required. Output only.

3.3.3 xxclr - Vector scalar clear

xxclr vst

Clear vector scalar register to 0.
Summary: vst = 0

Type code.
No implicit register effects.

Parameters **vst** (*VectorScalarRegister*) – Register to clear. Required. Output only.

3.4 Math

3.4.1 addi. - Add immediate and record

addi. rt ra si

Add immediate value to general register.
Record comparison with 0.
Summary: rt = ra + si

Type code.

Register effects
cr0 - Compare with 0 result

Parameters

- **rt** (*GeneralRegister*) – Receives result. Required. Output only.
- **ra** (*0|GeneralRegister*) – Variable addend. Required. Input only.
- **si** (*SignedInteger16b*) – Constant addend. Required. Input only.

3.4.2 dec. - Decrement and record

dec. rt ra

Decrement general register by amount in general register.
Record comparison with 0.
Summary: rt -= ra

Type code.

Register effects
cr0 - Compare with 0 result

Parameters

- **rt** (*GeneralRegister*) – Register to decrement. Required. Input and output.
- **ra** (*GeneralRegister*) – Amount to decrement. Required. Input only.

3.4.3 dec - Decrement

dec rt ra

Decrement general register by amount in general register.
Summary: rt -= ra

Type code.
No implicit register effects.

Parameters

- **rt** (*GeneralRegister*) – Register to decrement. Required. Input and output.
- **ra** (*GeneralRegister*) – Amount to decrement. Required. Input only.

3.4.4 dec1 - Decrement by 1

dec1 rt

Decrement general register by 1.
Summary: rt -= 1

Type code.
No implicit register effects.

Parameters **rt** (*0|GeneralRegister*) – Register to decrement. Required. Input and output.

Raises **Error** – If rt is r0.

3.4.5 deci - Decrement immediate

deci rt amount

Decrement general register by immediate value.
Summary: rt -= amount

Type code.
No implicit register effects.

Parameters

- **rt** (*0|GeneralRegister*) – Register to decrement. Required. Input and output.
- **amount** (*SignedInteger16b*) – Amount to decrement. Required. Input only.

Raises **Error** – If rt is r0.

3.4.6 inc. - Increment and record

inc. rt ra

Increment general register by value in general register.
Record comparison with 0.
Summary: rt += ra

Type code.

Register effects
cr0 - Compare with 0 result

Parameters

- **rt** (*GeneralRegister*) – Register to increment. Required. Input and output.
- **ra** (*GeneralRegister*) – Amount to increment. Required. Input only.

3.4.7 inc - Increment

inc rt ra

Increment general register by amount in general register.
Summary: rt += ra

Type code.
No implicit register effects.

Parameters

- **rt** (*GeneralRegister*) – Register to increment. Required. Input and output.
- **ra** (*GeneralRegister*) – Amount to increment. Required. Input only.

3.4.8 inc1 - Increment by 1

inc1 rt

Increment general register by 1.
Summary: rt += 1

Type code.
No implicit register usage.

Parameters **rt** (*0|GeneralRegister*) – Register to increment. Required. Input and output.

Raises **Error** – If rt is r0.

3.4.9 inci - Increment immediate

inci rt amount

Increment general register by immediate value.
Summary: rt += amount

Type code.
No implicit register usage.

Parameters

- **rt** (*0|GeneralRegister*) – Register to increment. Required. Input and output.
- **amount** (*SignedInteger16b*) – Amount to increment. Required. Input only.

Raises **Error** – If rt is r0.

3.4.10 subi. - Subtract immediate and record

subi. rt ra si

Subtract immediate value from general register.
Record comparison with 0.
Summary: rt = ra - si

Type code.

Register effects
cr0 - Compare with 0 result

Parameters

- **rt** (*GeneralRegister*) – Receives result. Required. Output only.
- **ra** (*0|GeneralRegister*) – Variable subtrahend. Required. Input only.
- **si** (*SignedInteger16b*) – Constant minuend. Required. Input only.

CHAPTER 4

Symbols

4.1 File Descriptor

4.1.1 stdin - Standard input

Standard input stream. Value 0.

```
stdin
```

4.1.2 stdout - Standard output

Standard output stream. Value 1.

```
stdout
```

4.1.3 stderr - Standard error

Standard error stream. Value 2.

```
stderr
```

4.2 Register

4.2.1 r0-31 - General

Names for general purpose registers.
Provides all general registers, r0-31.

```
r0 r1 r2 r3 r4 r5 r6 r7
```

```
r8 r9 r10 r11 r12 r13 r14 r15
r16 r17 r18 r19 r20 r21 r22 r23
r24 r25 r26 r27 r28 r29 r30 r31
```

4.2.2 f0-31 - Floating point

Names for floating point registers.
Provides all floating point registers, f0-31.

```
f0 f1 f2 f3 f4 f5 f6 f7
f8 f9 f10 f11 f12 f13 f14 f15
f16 f17 f18 f19 f20 f21 f22 f23
f24 f25 f26 f27 f28 f29 f30 f31
```

4.2.3 v0-31 - Vector

Names for vector registers.
Provides all vector registers, v0-31.

```
v0 v1 v2 v3 v4 v5 v6 v7
v8 v9 v10 v11 v12 v13 v14 v15
v16 v17 v18 v19 v20 v21 v22 v23
v24 v25 v26 v27 v28 v29 v30 v31
```

4.2.4 vs0-63 - Vector scalar

Names for vector scalar registers.
Provides all vector scalar registers, vs0-63.

```
vs0 vs1 vs2 vs3 vs4 vs5 vs6 vs7
vs8 vs9 vs10 vs11 vs12 vs13 vs14 vs15
vs16 vs17 vs18 vs19 vs20 vs21 vs22 vs23
vs24 vs25 vs26 vs27 vs28 vs29 vs30 vs31
vs32 vs33 vs34 vs35 vs36 vs37 vs38 vs39
vs40 vs41 vs42 vs43 vs44 vs45 vs46 vs47
vs48 vs49 vs50 vs51 vs52 vs53 vs54 vs55
vs56 vs57 vs58 vs59 vs60 vs61 vs62 vs63
```

4.2.5 cr0-7 - Condition field

Names for condition register fields.
Provides all condition register fields, cr0-7.

```
cr0 cr1 cr2 cr3 cr4 cr5 cr6 cr7
```