Stack Processor

Generated by Doxygen 1.8.10

Fri Oct 23 2015 13:40:29

Contents

Chapter 1

Todo List

globalScope> Global CPU_construct (CPU *This)

Dynamic memory management, so its OK that only true is returned by now.

globalScope> Global stack_construct (Stack *This)

Dynamic memory management, so its OK that only true is returned by now

Chapter 2

Data Structure Index

2.1 Data Structures

Here are the data structures with brief descriptions:

Chapter 3

File Index

3.1 File List

Here is a list of all documented files with brief descriptions:

Chapter 4

Data Structure Documentation

4.1 CPU Struct Reference

Simple stack processor.

`#include <CPU.h>`

Data Fields

- TYPE rax
- Stack stack
- bool state

4.1.1 Detailed Description

Simple stack processor.

Processor is a structure that containes a stack and a register to operate with.

4.1.2 Field Documentation

4.1.2.1 TYPE CPU::rax

Register (64-bit).

4.1.2.2 Stack CPU::stack

Operating stack.

4.1.2.3 bool CPU::state

State of the CPU. true if ON, false if OFF.

The documentation for this struct was generated from the following file:

- CPU.h

4.2 Stack Struct Reference

Simple stack of integer.

`#include <stack.h>`

Data Fields

- TYPE values [MAX_SIZE]
- TYPE * top
- bool state

4.2.1 Detailed Description

Simple stack of integer.

Stack contains integer numbers and provides some operations with them

4.2.2 Field Documentation

4.2.2.1 bool Stack::state

State of the stack. true if valid, false otherwise.

4.2.2.2 TYPE* Stack::top

Pointer to the top element of stack.

4.2.2.3 TYPE Stack::values[MAX_SIZE]

Array of values.

The documentation for this struct was generated from the following file:

- stack.h

Chapter 5

File Documentation

5.1 CPU.h File Reference

```
#include "CPU.h"
#include <assert.h>
#include <errno.h>
```

Data Structures

- struct CPU

  Simple stack processor.

Functions

- bool CPU_construct (CPU *This)

  Standard CPU constructor.
- bool CPU_construct_copy (CPU *This, const CPU *other)

  Copy CPU constructor.
- void CPU_destruct (CPU *This)

  Destructs the CPU.
- bool CPU_OK (const CPU *This)

  Validates the CPU.
- void CPU_dump_ (const CPU *This, const char name[])

  Prints CPU's dump.
- bool CPU_push (CPU *This, const TYPE *register_)

  Pushes value to CPU stack.
- bool CPU_pop (CPU *This, TYPE *register_)

  Pops value from the CPU.
- bool CPU_add (CPU *This)

  Summs the top two elements of the stack.
- bool CPU_sub (CPU *This)

  Subtracts the penult stack element from the top one.
- bool CPU_mul (CPU *This)

  Multiplies the top two elements of the stack.
- bool CPU_div (CPU *This)

  Divides the top stack element by the previous.
- bool CPU_pow (CPU *This)

  Raise the top element in the power of the penult one.

5.1.1 Function Documentation

5.1.1.1 bool CPU_add (CPU * *This*)

Summs the top two elements of the stack.

Add the top element of stack to the previous and stroes the result in stack. Both top two elements are removed.

Parameters

This	Pointer to the CPU to perform operation on.

Returns

true if success, false otherwise. In case of fail invalidates CPU.

Warning

Stack must contain at least two elements.

5.1.1.2 bool CPU_construct (CPU * *This*)

Standard CPU constructor.

Constructs CPU with empty CPU and register.

Parameters

This	Pointer to the CPU to be constructed.

Returns

1 (true) if success, 0 (false) otherwise.

Todo Dynamic memory management, so its OK that only true is returned by now.

5.1.1.3 bool CPU_construct_copy (CPU * *This*, const CPU * *other*)

Copy CPU constructor.

Constructs CPU as copy of other. Writes to errno.

Parameters

This	Pointer to the CPU to be constructed.
other	The CPU to copy from.

Returns

1 (true) if success, 0 (false) otherwise.

5.1.1.4 void CPU_destruct (CPU * *This*)

Destructs the CPU.

Destructs the CPU, setting its values to poison.

Parameters

This	Pointer to the CPU to be destructed.

5.1.1.5 bool CPU_div (CPU * *This*)

Divides the top stack element by the previous.

Gets the top element of stack and divides it by the penult stack element. Both top two elements are removed and the result is written to stack.

Parameters

This	Pointer to the CPU to perform operation on.

Returns

true if success, false otherwise. In case of fail invalidates CPU.

Warning

Stack must contain at least two elements.

5.1.1.6 void CPU_dump_ (const CPU * *This*, const char *name[ ]*)

Prints CPU's dump.

Outputs the current state of CPU.

Parameters

This	Pointer to the CPU to be dumped.

5.1.1.7 bool CPU_mul (CPU * *This*)

Multiplies the top two elements of the stack.

Multiplies the top element of stack to the previous and stroes the result in stack. Both top two elements are removed.

Parameters

This	Pointer to the CPU to perform operation on.

Returns

true if success, false otherwise. In case of fail invalidates CPU.

Warning

Stack must contain at least two elements.

5.1.1.8 bool CPU_OK (const CPU * *This*)

Validates the CPU.

Checks if the CPU is correct according to its values.

Parameters

This	Pointer to the CPU to be checked.

Returns

true if the CPU is valid, false otherwise.

5.1.1.9 bool CPU_pop (CPU * *This*, TYPE * *register_*)

Pops value from the CPU.

Pops the top element from the CPU and returnes it.

Parameters

This	Pointer to the CPU to perform operation on.
*register_*	Register where popped value is stored.

Returns

true if success, false otherwise. In case it wasn't successful, invalidates CPU.

Warning

Stack must contain at least one element.

5.1.1.10 bool CPU_pow (CPU * *This*)

Raise the top element in the power of the penult one.

Gets the top element of stack and raises it to power equal to the penult stack element. Both top two elements are removed and the result is written to stack.

Parameters

This	Pointer to the CPU to perform operation on.

Returns

true if success, false otherwise. In case of fail invalidates CPU.

Warning

Stack must contain at least two elements.

5.1.1.11 bool CPU_push (CPU * *This*, const TYPE * *register_*)

Pushes value to CPU stack.

Puts the value from given register at the top of CPU stack.

Parameters

This	Pointer to the CPU to perform operation on.
*register_*	Address of the register where value is stored.

Returns

true if success, false otherwise.

5.1.1.12 bool CPU_sub (CPU * *This*)

Subtracts the penult stack element from the top one.

Gets the top element of stack and subtracts the penult stack element from it. Both top two elements are removed and the result is written to stack.

Parameters

This	Pointer to the CPU to perform operation on.

Returns

true if success, false otherwise. In case of fail invalidates CPU.

Warning

Stack must contain at least two elements.

5.2 stack.h File Reference

```
#include <assert.h>
#include <errno.h>
#include <string.h>
```

Data Structures

- struct Stack

  Simple stack of integer.

Macros

- #define TYPE int

  Type of stack's values.
- #define MAX_SIZE 16

  Maximum number of elements in stack.
- #define **MAX_PRINTED** MAX_SIZE
- #define Stack_dump(This) stack_dump_(This, #This)

  More comfortable dump.
- #define stack_dump(This) Stack_dump(This)

  To be stylish.
- #define **Stack_OK**(This) stack_OK(This)

Functions

- bool stack_construct (Stack *This)

  Standard stack constructor.
- bool stack_construct_copy (Stack *This, const Stack *other)

  Copy stack constructor.
- void stack_destruct (Stack *This)

  Destructs the stack.
- bool Stack_OK (const Stack *This)

  Validates the stack.
- void stack_dump_ (const Stack *This, const char name[])

  Prints stack's dump.
- bool stack_push (Stack *This, TYPE value)

  Pushes value to stack.
- TYPE stack_pop (Stack *This)

  Pops value from the stack.

5.2.1 Function Documentation

5.2.1.1 bool stack_construct (Stack * *This*)

Standard stack constructor.

Constructs stack with the top pointer at NULL. Writes to errno.

Parameters

This	Pointer to the stack to be constructed.

Returns

1 (true) if success, 0 (false) otherwise.

Todo Dynamic memory management, so its OK that only true is returned by now

5.2.1.2 bool stack_construct_copy (Stack * *This*, const Stack * *other*)

Copy stack constructor.

Constructs stack as copy of other. Writes to errno.

Parameters

This	Pointer to the stack to be constructed.
other	The stack to copy from.

Returns

1 (true) if success, 0 (false) otherwise.

5.2.1.3 void stack_destruct (Stack * *This*)

Destructs the stack.

Destructs the stack, setting its values to poison.

Parameters

This	Pointer to the stack to be destructed.

5.2.1.4 void stack_dump_ (const Stack * *This,* const char *name[ ]*)

Prints stack's dump.

Outputs the current state of stack.

Parameters

This	Pointer to the stack to be dumped.

5.2.1.5 bool Stack_OK (const Stack * *This*)

Validates the stack.

Checks if the stack is correct according to its values.

Parameters

This	Pointer to the stack to be checked.

Returns

true if the stack is valid, false otherwise.

5.2.1.6 TYPE stack_pop (Stack * *This*)

Pops value from the stack.

Pops the top element from the stack and returnes it.

Parameters

This	Pointer to the stack to perform operation on.

Returns

Value that was popped. In case it wasn't successful, invalidates stack.

5.2.1.7 bool stack_push (Stack * *This,* TYPE *value*)

Pushes value to stack.

Put the given value to stack as top element.

Parameters

This	Pointer to the stack to perform operation on.
value	Value to be pushed.

Returns

true if success, false otherwise

Index