C8E

0.1

Generated by Doxygen 1.7.2

Wed Dec 15 2010 19:05:15

Contents

Chapter 1

File Index

1.1 File List

Here is a list of all files with brief descriptions:

Chapter 2

File Documentation

2.1 src/C8E.c File Reference

```
#include "Logs.h"
#include "Memory.h"
#include "CartridgeReader.h"
#include <GLUT/glut.h>
```

Defines

- #define IDLE_TIME 16
- #define SCREEN_WIDTH 64
- #define SCREEN_HEIGTH 32

Functions

- void render ()
- void CPUTick (int t)
- void setupGlut (int argc, char **argv)
- int main (int argc, char **argv)

2.1.1 Define Documentation

2.1.1.1 #define IDLE_TIME 16

Definition at line 11 of file C8E.c.

2.1.1.2 #define SCREEN_HEIGTH 32

Definition at line 14 of file C8E.c.

2.1.1.3 #define SCREEN_WIDTH 64

Definition at line 13 of file C8E.c.

2.1.2 Function Documentation

2.1.2.1 void CPUTick (int *t*)

Definition at line 34 of file C8E.c.

2.1.2.2 int main (int *argc,* char ** *argv*)

Definition at line 48 of file C8E.c.

2.1.2.3 void render ( )

Definition at line 16 of file C8E.c.

2.1.2.4 void setupGlut (int *argc,* char ** *argv*)

Definition at line 41 of file C8E.c.

2.2 src/CartridgeReader.c File Reference

```
#include "CartridgeReader.h"
#include <stdio.h>
```

Functions

- int readCartridge (const char *const filename, char *data)
  Provide a pointer to the cardridge data.

2.2.1 Function Documentation

2.2.1.1 int readCartridge (const char *const *filename,* char * *data*)

Provide a pointer to the cardridge data.

Parameters

in	*filename*	Point the file to load into memory
out	*data*	Buffer that eventually receive the cardridge data. It must be initialized and big enougth.

Returns

0 if the file exists, 1 otherwise.

Definition at line 31 of file CartridgeReader.c.

2.3 src/CartridgeReader.h File Reference

Define all functions, variables and defines for cartridge management.

Functions

- int readCartridge (const char ∗const filename, char ∗data)

  Provide a pointer to the cardridge data.

2.3.1 Detailed Description

Define all functions, variables and defines for cartridge management.

Version

0.1

Date

December 12, 2010

Author

Maxime Gaudin

Definition in file CartridgeReader.h.

2.3.2 Function Documentation

2.3.2.1 int readCartridge (const char ∗const *filename,* char ∗ *data*)

Provide a pointer to the cardridge data.

Parameters

in	*filename*	Point the file to load into memory
out	*data*	Buffer that eventually receive the cardridge data. It must be initialized and big enougth.

Returns

0 if the file exists, 1 otherwise.

Definition at line 31 of file CartridgeReader.c.

2.4 src/CPU.c File Reference

```
#include "CPU.h"
#include "Memory.h"
#include "Logs.h"
```

Functions

- int setupCPU ()
- void cleanupCPU ()
- void tick ()
- void handleOpCode ()

2.4.1 Detailed Description

Version

0.1

Date

December 13, 2010

Author

Maxime Gaudin

Definition in file CPU.c.

2.4.2 Function Documentation

2.4.2.1 void cleanupCPU ( )

Definition at line 69 of file CPU.c.

2.4.2.2 void handleOpCode ( )

Definition at line 289 of file CPU.c.

2.4.2.3 int setupCPU ( )

Definition at line 52 of file CPU.c.

2.4.2.4 void tick ( )

Definition at line 284 of file CPU.c.

2.5 src/CPU.h File Reference

Defines

- #define MAX_STACK_SIZE 0xF

  Define the maximum stack size, i.e. the maximum amount of subroutine calls.

Functions

- int setupCPU ()
- void cleanupCPU ()
- void tick ()

2.5.1 Detailed Description

Version

0.1

Date

December 12, 2010

Author

Maxime Gaudin

Definition in file CPU.h.

2.5.2 Define Documentation

2.5.2.1 #define MAX_STACK_SIZE 0xF

Define the maximum stack size, i.e. the maximum amount of subroutine calls.

Definition at line 31 of file CPU.h.

2.5.3 Function Documentation

2.5.3.1 void cleanupCPU ( )

Definition at line 69 of file CPU.c.

2.5.3.2 int setupCPU ( )

Definition at line 52 of file CPU.c.

2.5.3.3 void tick ( )

Definition at line 284 of file CPU.c.

2.6 src/Logs.c File Reference

```
#include "Logs.h"
```

Functions

- int setupLogs (int redirect, unsigned char debugLevel, char ∗const outputFilename)

  Setup output log file and debug level to values passed in paramaters. Moreover, a file descriptor is created and initialized. if [redirect], log are also written in stdou.

- int closeLogs ()

  Close output log file descriptor and flush file buffer.

- void addEntry (unsigned char level, const char ∗const message)

  Add new entry in output log file if [level] is below or equal to debug level.

2.6.1 Function Documentation

2.6.1.1 void addEntry (unsigned char *level,* const char ∗const *message*)

Add new entry in output log file if [level] is below or equal to debug level.

Definition at line 56 of file Logs.c.

2.6.1.2 int closeLogs ( )

Close output log file descriptor and flush file buffer.

Returns

0 if success, 0 otherwise.

Definition at line 50 of file Logs.c.

2.6.1.3 int setupLogs (int *redirect,* unsigned char *debugLevel,* char ∗const *outputFilename*)

Setup output log file and debug level to values passed in paramaters. Moreover, a file descriptor is created and initialized. if [redirect], log are also written in stdou.

Returns

0 if success, 0 otherwise.

Definition at line 36 of file Logs.c.

2.7 src/Logs.h File Reference

```
#include <stdio.h>
```

Defines

- #define DEFAULT_DEBUG_LEVEL 1

  Specifies teh default debug level : Warning.

- #define DEFAULT_OUTPUT_FILENAME "DEBUG_LOGS"

  Specifies the default output filename, i.e. the file where log will be written.

Enumerations

- enum DEBUG_LEVELS {
  ERROR = 0, WARNING = 1, DRAWING = 2, DISASSEMBLY = 3,
  LOW_LEVEL_OPERATION = 4 }

Functions

- int setupLogs (int redirect, unsigned char debugLevel, char ∗const outputFilename)

  Setup output log file and debug level to values passed in paramaters. Moreover, a file descriptor is created and initialized. if [redirect], log are also written in stdou.

- int closeLogs ()

Close output log file descriptor and flush file buffer.

- void addEntry (unsigned char level, const char *const message)

  Add new entry in output log file if [level] is below or equal to debug level.

2.7.1 Define Documentation

2.7.1.1 #define DEFAULT_DEBUG_LEVEL 1

Specifies teh default debug level : Warning.

Definition at line 23 of file Logs.h.

2.7.1.2 #define DEFAULT_OUTPUT_FILENAME "DEBUG_LOGS"

Specifies the default output filename, i.e. the file where log will be written.

Definition at line 26 of file Logs.h.

2.7.2 Enumeration Type Documentation

2.7.2.1 enum DEBUG_LEVELS

Enumerator:

ERROR

WARNING

DRAWING

DISASSEMBLY

LOW_LEVEL_OPERATION

Definition at line 20 of file Logs.h.

2.7.3 Function Documentation

2.7.3.1 void addEntry (unsigned char *level,* const char *const *message*)

Add new entry in output log file if [level] is below or equal to debug level.

Definition at line 56 of file Logs.c.

2.7.3.2 int closeLogs ( )

Close output log file descriptor and flush file buffer.

Returns

0 if success, 0 otherwise.

Definition at line 50 of file Logs.c.

2.7.3.3 int setupLogs (int *redirect,* unsigned char *debugLevel,* char ∗const *outputFilename*)

Setup output log file and debug level to values passed in paramaters. Moreover, a file descriptor is created and initialized. if [redirect], log are also written in stdou.

Returns

0 if success, 0 otherwise.

Definition at line 36 of file Logs.c.

2.8 src/Memory.c File Reference

Define all functions, variables and defines for memory management.

```
#include "Memory.h"
#include "Logs.h"
#include <stdlib.h>
#include <string.h>
```

Functions

- int setupMemory ()
  Initialize memory to 0.
- void cleanupMemory ()
  Cleanup all memory.
- int write (unsigned short addr, char ∗const data, unsigned int len)
  write [len] bytes from [data] into memory at adress [addr]
- int read (short addr, unsigned short len, char ∗const buffer)
  Read [len] bytes of data from address [addr] to buffer.

2.8.1 Detailed Description

Define all functions, variables and defines for memory management.

Version

0.1

Date

December 12, 2010

Author

Maxime Gaudin

Definition in file Memory.c.

2.8.2 Function Documentation

2.8.2.1 void cleanupMemory ( )

Cleanup all memory.

Definition at line 39 of file Memory.c.

2.8.2.2 int read (short *addr,* unsigned short *len,* char *const *buffer*)

Read [len] bytes of data from address [addr] to buffer.

Parameters

in	*addr*	Address where read begins
in	*len*	Number of bytes read
out	*buffer*	Pointer to the data buffer

Returns

0 if success, 1 otherwise.

Definition at line 56 of file Memory.c.

2.8.2.3 int setupMemory ( )

Initialize memory to 0.

Returns

0 if success, 1 otherwise.

Definition at line 28 of file Memory.c.

2.8.2.4 int write (unsigned short *addr,* char *const *data,* unsigned int *len*)

write [len] bytes from [data] into memory at adress [addr]

Parameters

in	*addr*	Address where data will be written
in	*data*	Pointer to data buffer
in	*len*	Number of byte written

Returns

0 if success, 1 otherwise.

Definition at line 44 of file Memory.c.

2.9 src/Memory.h File Reference

Defines

- #define RESERVED_MEMORY_START 0x0

  Specifies where memory starts (0x0, what a surprise isn't it ??).

- #define RESERVED_MEMORY_STOP 0x200

  Specifies where the memory stops.

- #define DATA_SPACE_START 0x200

  Specifies the beginning of the data space.

- #define DATA_SPACE_STOP 0xFFF

  Specifies the end of the data space.

- #define MAX_REGISTERS 0xF

  Specifies the maximum number of registers..

Functions

- int setupMemory ()

  Initialize memory to 0.

- void cleanupMemory ()

  Cleanup all memory.

- int write (unsigned short addr, char ∗const data, unsigned int len)

  write [len] bytes from [data] into memory at adress [addr]

- int read (short addr, unsigned short len, char ∗const buffer)

  Read [len] bytes of data from address [addr] to buffer.

2.9.1 Define Documentation

2.9.1.1 #define DATA_SPACE_START 0x200

Specifies the beginning of the data space.

Definition at line 36 of file Memory.h.

2.9.1.2 #define DATA_SPACE_STOP 0xFFF

Specifies the end of the data space.

Definition at line 38 of file Memory.h.

2.9.1.3 #define MAX_REGISTERS 0xF

Specifies the maximum number of registers..

Definition at line 41 of file Memory.h.

2.9.1.4 #define RESERVED_MEMORY_START 0x0

Specifies where memory starts (0x0, what a surprise isn't it ??).

Definition at line 31 of file Memory.h.

2.9.1.5 #define RESERVED_MEMORY_STOP 0x200

Specifies where the memory stops.

Definition at line 33 of file Memory.h.

2.9.2 Function Documentation

2.9.2.1 void cleanupMemory ( )

Cleanup all memory.

Definition at line 39 of file Memory.c.

2.9.2.2 int read (short *addr,* unsigned short *len,* char *const *buffer*)

Read [len] bytes of data from address [addr] to buffer.

Parameters

in	*addr*	Address where read begins
in	*len*	Number of bytes read
out	*buffer*	Pointer to the data buffer

Returns

0 if success, 1 otherwise.

Definition at line 56 of file Memory.c.

2.9.2.3 int setupMemory ( )

Initialize memory to 0.

Returns

0 if success, 1 otherwise.

Definition at line 28 of file Memory.c.

2.9.2.4 int write (unsigned short *addr,* char *const *data,* unsigned int *len*)

write [len] bytes from [data] into memory at adress [addr]

Parameters

in	*addr*	Address where data will be written
in	*data*	Pointer to data buffer
in	*len*	Number of byte written

Returns

0 if success, 1 otherwise.

Definition at line 44 of file Memory.c.

Index