Ape

0.0.1

Generated by Doxygen 1.8.14

Contents

Chapter 1

Main Page

Welcome

Ape (Another PC Emulator) is an experimental IBM PC compatible emulator written in C++17.

Chapter 2

Namespace Index

2.1 Namespace List

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

Chapter 3

Hierarchical Index

3.1 Class Hierarchy

This inheritance list is sorted roughly, but not completely, alphabetically:

Chapter 4

Class Index

4.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

Chapter 5

File Index

5.1 File List

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

Chapter 6

Namespace Documentation

6.1 Core::CPU Namespace Reference

Representation of the Central Processing Unit.

Classes

- struct Breakpoint
- class CPUException

  Base class thrown by every CPU related exception.
- union GPR
- class Instruction

  High-Level representation of a instruction.
- class InvalidInstructionException
- class InvalidParameterException

  Thrown when the CPU hits an invalid mod byte that can't be decoded.
- class ParameterLengthMismatchException
- class UnhandledInstructionException

  Thrown when the CPU hits an unimplemented Instruction::Type.
- class UnhandledInterruptException

  Thrown when a interrupt is neither handled by software nor hardware.
- class UnhandledParameterException

  Thrown when the CPU hits an unimplemented Instruction::Parameter::Type.
- class UnsupportedParameterException

  Thrown when a instruction doesn't support the parameter type provided.

Typedefs

- using **StateCallbackFunc** = std::function< void(State)>

Enumerations

- enum **RepeatMode** : u8 { **None**, **Repeat**, **Repeat_Zero**, **Repeat_Non_Zero** }
- enum **State** : u8 { **Stopped**, **Running**, **Paused** }

Functions

- void **AddBreakpoint** (Breakpoint b)
- void **RemoveBreakpoint** (Breakpoint b)
- bool **IsBreakpointHit** ()
- bool **IsBreakpoint** (u16 segment, u16 offset)
- void Stop ()

  Stop the CPU.
- void **SetPaused** (bool value)
- bool **IsRunning** ()
- bool **IsPaused** ()
- State **GetState** ()
- void **RegisterStateChangedCallback** (StateCallbackFunc fnc)
- template<typename T , typename... U>
  size_t **GetAddress** (std::function< T(U...)> f)
- void **UnregisterStateChangedCallback** (StateCallbackFunc fnc)
- void **TriggerCallbacks** ()
- bool **HandleRepetition** ()
- void Tick ()

  Execute one CPU cycle.
- void Start ()

  Execute instructions until shutdown is requested.
- u16 **PrefixToValue** (Instruction::SegmentPrefix prefix)
- PRIVATE void **UpdateZF** (u16 value)
- void **UpdatePF** (u16 value)
- void **UpdateSF** (i16 value)
- template<typename T >
  void **UpdateOF** (i32 value)
- template<typename T >
  void **UpdateCF** (i32 value)
- void **CallInterrupt** (u8 vector)
- bool **CallBIOSInterrupt** (u8 vector)
- bool **CallMSDOSInterrupt** (u8 vector)
- std::string TypeToString (const Instruction::Type &type)

  Get the corresponding nmoroic for the Type provided.
- std::string ParameterTypeToString (const Instruction::Parameter::Type &type, Instruction::SegmentPrefix prefix=Instruction::SegmentPrefix::None)

  Get a human readable form of the Parameter::Type provided.
- bool ParameterNeedsResolving (const Instruction::Parameter::Type &parameter)

  Checks whether this Parameter::Type needs resolving.

Variables

- std::vector< Breakpoint > **breakpoints**
- GPR **A**
- GPR **B**
- GPR **C**
- GPR **D**
- u16 & AX = A.X

  AX (Accumulator)
- u16 & BX = B.X

  BX.
- u16 & CX = C.X

CX.

- u16 & DX = D.X

  DX.
- u8 & AH = A.b8.H

  AH (High)
- u8 & AL = A.b8.L

  AL (Low)
- u8 & BH = B.b8.H

  BH (High)
- u8 & BL = B.b8.L

  BL (Low)
- u8 & CH = C.b8.H

  CH (High)
- u8 & CL = C.b8.L

  CL (Low)
- u8 & DH = D.b8.H

  DH (High)
- u8 & DL = D.b8.L

  DL (Low)
- u16 IP = 0

  Instruction Pointer.
- u16 DI = 0

  Destination Index.
- u16 SI = 0

  Source Index.
- u16 SP = 0

  Stack Pointer.
- u16 BP = 0

  Base Pointer.
- u16 CS = 0

  Code Segment.
- u16 DS = 0

  Data Segment.
- u16 SS = 0

  Stack Segment.
- u16 ES = 0

  Extra(?) Segment.
- bool AF = false

  Adjust Flag.
- bool CF = false

  Carry Flag.
- bool IF = false

  Interrupt Flag.
- bool DF = false

  Direction Flag.
- bool OF = false

  Overflow Flag.
- bool PF = false

  Parity Flag.
- bool SF = false

  Sign Flag.

- bool ZF = false
  Zero Flag.
- bool simulate_msdos = false
  Simulate MS-DOS (Handle its interrupts)
- bool pause_on_boot = false
- std::atomic< bool > **running**
- std::atomic< bool > **paused**
- u64 **clock_speed** = 5'000'000
- RepeatMode **s_repeat_mode** = RepeatMode::None
- std::vector< StateCallbackFunc > **fncs**
- Breakpoint **just_hit** = {0, 0}

6.1.1 Detailed Description

Representation of the Central Processing Unit.

6.1.2 Variable Documentation

6.1.2.1 pause_on_boot

```
bool Core::CPU::pause_on_boot = false
```

Whether or not to pause after emulation has started (Useful for debugging purposes)

6.2 Core::HW::FloppyDrive Namespace Reference

Representation of a floppy drive.

Functions

- bool Insert (const std::string &path)
  Insert an image into the drive.
- bool HasDisc ()
  Check if a disc is present.
- u32 GetSize ()
  Get the size of the inserted disc.
- bool **GuessFormat** ()
- bool IsBootable ()
  Check if the provided image is bootable.
- bool Read (u32 offset, u32 size, u8 *buffer)
  Read data from the image.
- bool **Read** (u8 cylinder, u8 head, u8 sector, u8 count, u8 *buffer)
- void Eject ()
  Eject the image.
- u32 GetSectorSize ()
  Get size of a floppy.
- u32 GetSectorsPerTrack ()
  Get sectors per track.
- u32 GetHeadCount ()
  Get head count.

Variables

- u16 **m_sectors_per_track**
- u16 **m_sector_size**
- u16 **m_head_count**
- std::unique_ptr< std::ifstream > **m_file**

6.2.1 Detailed Description

Representation of a floppy drive.

6.3 Core::Machine Namespace Reference

Representation of a PC.

Functions

- void **Init** ()
- bool BootFloppy ()
  Boot the machine from the floppy drive.
- void Stop ()
  Stop the machine.
- void Pause ()
  Pause the machine (Or unpause it if it's paused already)
- bool BootCOM (const std::string &file, const std::string &&parameters="")
  Directly execute a COM file.

6.3.1 Detailed Description

Representation of a PC.

6.4 Core::Memory Namespace Reference

Wrapper around emulated RAM.

Functions

- std::vector< u8 > & Get ()
  Get the contents of RAM.
- u32 VirtToPhys (u16 segment, u16 offset)
  Converts a virtual address to an absolute one.
- template<typename T >
  T & **Get** (u16 segment, u16 offset)
- template<typename T >
  T * **GetPtr** (u16 segment, u16 offset)

6.4.1 Detailed Description

Wrapper around emulated RAM.

Chapter 7

Class Documentation

7.1 Core::CPU::Breakpoint Struct Reference

Public Attributes

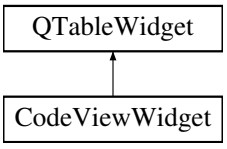
- u16 **segment**
- u16 **offset**

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

- /home/max/sources/ape/Source/Core/CPU/Breakpoint.h

7.2 CodeViewWidget Class Reference

Inheritance diagram for CodeViewWidget:



Public Member Functions

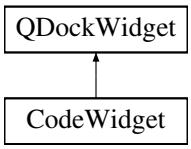
- u16 **GetSegment** () const
- u16 **GetOffset** () const
- void **SetSegment** (u16 segment)
- void **SetOffset** (u16 offset)
- void **resizeEvent** (QResizeEvent ∗) override

The documentation for this class was generated from the following files:

- /home/max/sources/ape/Source/ApeQt/Debugger/CodeViewWidget.h
- /home/max/sources/ape/Source/ApeQt/Debugger/CodeViewWidget.cpp

7.3 CodeWidget Class Reference

Inheritance diagram for CodeWidget:



Signals

- void **Closed** ()

Public Member Functions

- void **closeEvent** (QCloseEvent ∗) override

The documentation for this class was generated from the following files:

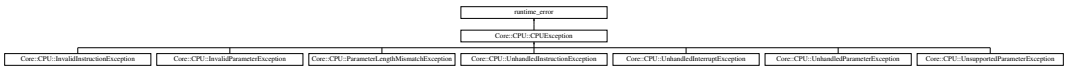
- /home/max/sources/ape/Source/ApeQt/Debugger/CodeWidget.h
- /home/max/sources/ape/Source/ApeQt/Debugger/CodeWidget.cpp

7.4 Core::CPU::CPUException Class Reference

Base class thrown by every CPU related exception.

`#include <Exception.h>`

Inheritance diagram for Core::CPU::CPUException:



Protected Member Functions

- **CPUException** (const std::string &message)

7.4.1 Detailed Description

Base class thrown by every CPU related exception.

The documentation for this class was generated from the following files:

- /home/max/sources/ape/Source/Core/CPU/Exception.h
- /home/max/sources/ape/Source/Core/CPU/Exception.cpp

7.5 Core::CPU::GPR Union Reference

Public Attributes

- u16 **X** = 0
- struct {
  u8 **L**
  u8 **H**
  } **b8**

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

- /home/max/sources/ape/Source/Core/CPU/CPU.h

7.6 Core::CPU::Instruction Class Reference

High-Level representation of a instruction.

`#include <Instruction.h>`

Classes

- class Parameter
  High-Level representation of a instruction parameter.

Public Types

- enum SegmentPrefix : u8
  Enum of all possible segment prefixes.
- enum Type : u8 { **PRIVATE** }
  Determines the type of the instruction.

Public Member Functions

- Instruction (u8 opcode, u32 offset=0)
  Turns the provided opcode into an Instruction.
- Instruction (const Instruction &ins, u8 opcode, u32 offset=0)
  One but with prefixes.
- Type GetType () const
  Get the Type associated with this instruction.
- SegmentPrefix GetPrefix () const
  Get the SegmentPrefix associated with this instruction.
- bool IsResolved ()
  Checks whether this Instruction needs further resolving.
- bool IsPrefix () const

Checks if this instruction is actually a prefix.

- bool Resolve (u8 mod, std::vector< u8 > data)

  Resolves the Instruction.
- u8 GetLength (u8 mod)

  Get the length of the instruction provided.
- std::string ToString () const

  Get a disassembly for the Instruction provided.
- const std::vector< Parameter > & GetParameters () const

  Get a vector of parameters.
- Parameter & GetParameter (size_t index)

  Get a specific vector.
- void AddParameter (Parameter parameter)

  Add a new parameter.

7.6.1 Detailed Description

High-Level representation of a instruction.

7.6.2 Member Enumeration Documentation

7.6.2.1 SegmentPrefix

```
enum Core::CPU::Instruction::SegmentPrefix : u8 [strong]
```

Enum of all possible segment prefixes.

These prefixes override the default segment of an instruction.

7.6.3 Constructor & Destructor Documentation

7.6.3.1 Instruction()

```
Core::CPU::Instruction::Instruction (
            u8 opcode,
            u32 offset = 0 ) [explicit]
```

Turns the provided opcode into an Instruction.

Parameters

opcode	Opcode to be decoded

7.6.4 Member Function Documentation

7.6.4.1 GetLength()

```
u8 Instruction::GetLength (
            u8 mod )
```

Get the length of the instruction provided.

Returns

Length of the instruction in bytes

7.6.4.2 Resolve()

```
bool Instruction::Resolve (
            u8 mod,
            std::vector< u8 > data )
```

Resolves the Instruction.

Parameters

mod	Modifier byte. The byte after the opcode regardless if it is used as such
data	needs to have as many bytes as specified in GetInstructionLength()

Returns

Returns `false` if there was an error during resolving.

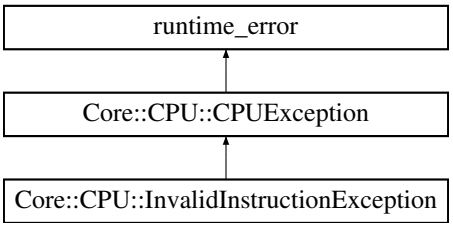
The documentation for this class was generated from the following files:

- /home/max/sources/ape/Source/Core/CPU/Instruction.h
- /home/max/sources/ape/Source/Core/CPU/Decoder.cpp
- /home/max/sources/ape/Source/Core/CPU/Instruction.cpp

7.7 Core::CPU::InvalidInstructionException Class Reference

```
#include <Exception.h>
```

Inheritance diagram for Core::CPU::InvalidInstructionException:



Public Member Functions

- **InvalidInstructionException** (u8 opcode)

Additional Inherited Members

7.7.1 Detailed Description

Thrown when the CPU hits an instruction that couldn't be resolved / is invalid

The documentation for this class was generated from the following files:

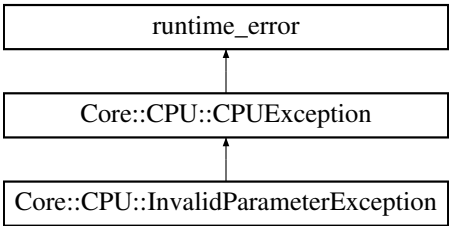
- /home/max/sources/ape/Source/Core/CPU/Exception.h
- /home/max/sources/ape/Source/Core/CPU/Exception.cpp

7.8 Core::CPU::InvalidParameterException Class Reference

Thrown when the CPU hits an invalid mod byte that can't be decoded.

`#include <Exception.h>`

Inheritance diagram for Core::CPU::InvalidParameterException:



Public Member Functions

- **InvalidParameterException** (u8 opcode, u8 mod)

Additional Inherited Members

7.8.1 Detailed Description

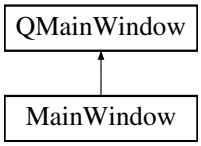
Thrown when the CPU hits an invalid mod byte that can't be decoded.

The documentation for this class was generated from the following files:

- /home/max/sources/ape/Source/Core/CPU/Exception.h
- /home/max/sources/ape/Source/Core/CPU/Exception.cpp

7.9 MainWindow Class Reference

Inheritance diagram for MainWindow:



Public Member Functions

- **MainWindow** (const std::string &&path="", bool floppy=false)

The documentation for this class was generated from the following files:

- /home/max/sources/ape/Source/ApeQt/MainWindow.h
- /home/max/sources/ape/Source/ApeQt/MainWindow.cpp

7.10 Core::CPU::Instruction::Parameter Class Reference

High-Level representation of a instruction parameter.

`#include <Instruction.h>`

Public Types

- enum Type : u8 { **PRIVATE** }

  Enum to determine the types of parameters.

Public Member Functions

- Parameter (Parameter::Type type)

  Create a parameter with the Instruction::Parameter::Type provided.
- void Resolve (u32 data)

  Provide the parameter with its missing data.
- void Resolve (Parameter::Type type, u32 data=0)

  Change the type of the parameter and provide missing data (if any)
- Type GetType () const

  Get the parameters type.
- template<typename T >
  T GetData () const

  Get the data associated with this parameter.
- bool IsResolved () const

  Returns `true` if this parameter is resolved.
- bool IsWord () const

  Returns `true` if this parameter points to or is a word.
- std::string ToString (SegmentPrefix prefix=SegmentPrefix::None, u32 offset=0) const

  Get a human readable form of this parameter.

7.10.1 Detailed Description

High-Level representation of a instruction parameter.

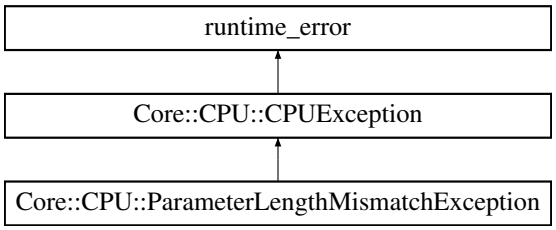
The documentation for this class was generated from the following files:

- /home/max/sources/ape/Source/Core/CPU/Instruction.h
- /home/max/sources/ape/Source/Core/CPU/Instruction.cpp

7.11 Core::CPU::ParameterLengthMismatchException Class Reference

`#include <Exception.h>`

Inheritance diagram for Core::CPU::ParameterLengthMismatchException:



Public Member Functions

- **ParameterLengthMismatchException** (const Instruction &ins, const Instruction::Parameter &p1, const Instruction::Parameter &p2)
- **ParameterLengthMismatchException** (const Instruction::Parameter &p)

Additional Inherited Members

7.11.1 Detailed Description

Thrown when a `word` and `byte` parameter are used together when they shouldn't be

The documentation for this class was generated from the following files:

- /home/max/sources/ape/Source/Core/CPU/Exception.h
- /home/max/sources/ape/Source/Core/CPU/Exception.cpp

7.12 ParameterParser Class Reference

Public Member Functions

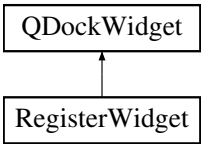
- void **AddCommand** (const std::string &name)
- void **AddFlag** (const std::string &name)
- void **AddString** (const std::string &name)
- bool **Parse** (int argc, char **argv)
- bool **CheckCommand** (const std::string &name)
- bool **CheckFlag** (const std::string &name)
- const std::string **GetString** (const std::string &name)

The documentation for this class was generated from the following files:

- /home/max/sources/ape/Source/Common/ParameterParser.h
- /home/max/sources/ape/Source/Common/ParameterParser.cpp

7.13 RegisterWidget Class Reference

Inheritance diagram for RegisterWidget:



Signals

- void **OnUpdate** ()
- void **Closed** ()

Public Member Functions

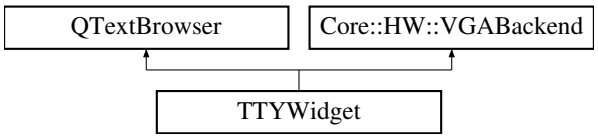
- void **Update** ()
- void **closeEvent** (QCloseEvent ∗) override

The documentation for this class was generated from the following files:

- /home/max/sources/ape/Source/ApeQt/Debugger/RegisterWidget.h
- /home/max/sources/ape/Source/ApeQt/Debugger/RegisterWidget.cpp

7.14 TTYWidget Class Reference

Inheritance diagram for TTYWidget:



Public Member Functions

- void **SetMode** (u8 mode) override
- void **Update** () override

The documentation for this class was generated from the following files:

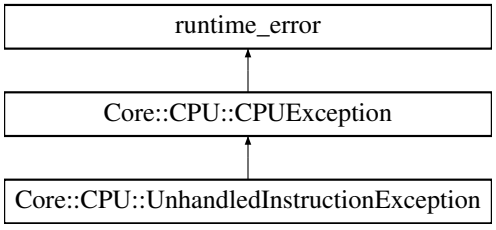
- /home/max/sources/ape/Source/ApeQt/TTYWidget.h
- /home/max/sources/ape/Source/ApeQt/TTYWidget.cpp

7.15 Core::CPU::UnhandledInstructionException Class Reference

Thrown when the CPU hits an unimplemented Instruction::Type.

`#include <Exception.h>`

Inheritance diagram for Core::CPU::UnhandledInstructionException:



Public Member Functions

- **UnhandledInstructionException** (const Instruction &ins)

Additional Inherited Members

7.15.1 Detailed Description

Thrown when the CPU hits an unimplemented Instruction::Type.

The documentation for this class was generated from the following files:

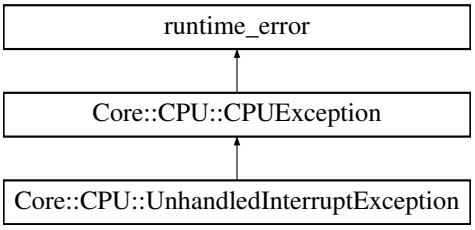
- /home/max/sources/ape/Source/Core/CPU/Exception.h
- /home/max/sources/ape/Source/Core/CPU/Exception.cpp

7.16 Core::CPU::UnhandledInterruptException Class Reference

Thrown when a interrupt is neither handled by software nor hardware.

`#include <Exception.h>`

Inheritance diagram for Core::CPU::UnhandledInterruptException:



Additional Inherited Members

7.16.1 Detailed Description

Thrown when a interrupt is neither handled by software nor hardware.

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

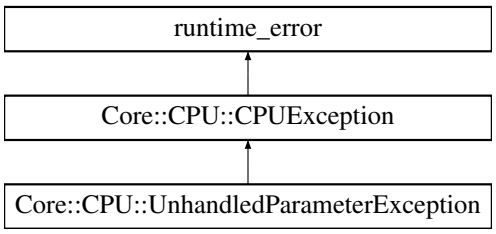
- /home/max/sources/ape/Source/Core/CPU/Exception.h

7.17 Core::CPU::UnhandledParameterException Class Reference

Thrown when the CPU hits an unimplemented Instruction::Parameter::Type.

`#include <Exception.h>`

Inheritance diagram for Core::CPU::UnhandledParameterException:



Public Member Functions

- **UnhandledParameterException** (const Instruction::Parameter &p)

Additional Inherited Members

7.17.1 Detailed Description

Thrown when the CPU hits an unimplemented Instruction::Parameter::Type.

The documentation for this class was generated from the following files:

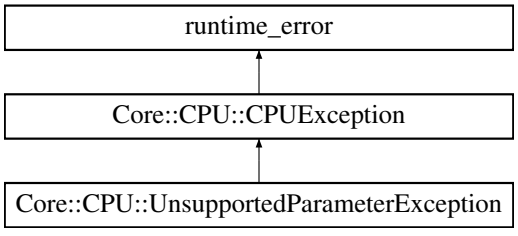
- /home/max/sources/ape/Source/Core/CPU/Exception.h
- /home/max/sources/ape/Source/Core/CPU/Exception.cpp

7.18 Core::CPU::UnsupportedParameterException Class Reference

Thrown when a instruction doesn't support the parameter type provided.

`#include <Exception.h>`

Inheritance diagram for Core::CPU::UnsupportedParameterException:



Public Member Functions

- **UnsupportedParameterException** (const Instruction &ins, const Instruction::Parameter &p)

Additional Inherited Members

7.18.1 Detailed Description

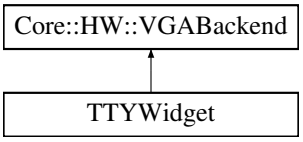
Thrown when a instruction doesn't support the parameter type provided.

The documentation for this class was generated from the following files:

- /home/max/sources/ape/Source/Core/CPU/Exception.h
- /home/max/sources/ape/Source/Core/CPU/Exception.cpp

7.19 Core::HW::VGABackend Class Reference

Inheritance diagram for Core::HW::VGABackend:



Public Member Functions

- virtual void **SetMode** (u8 mode)=0
- virtual void **Update** ()=0

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

- /home/max/sources/ape/Source/Core/HW/VGA.h

Chapter 8

File Documentation

8.1 /home/max/sources/ape/Source/Common/Logger.h File Reference

```
#include <string>
```

Macros

- #define LOG(msg) __MSG("LOG", __FILE__, __LINE__, msg)
  Log a message.
- #define WARN(msg) __MSG("WARNING", __FILE__, __LINE__, msg)
  Log a warning.
- #define ERROR(msg) __MSG("ERROR", __FILE__, __LINE__, msg)
  Log an error.

8.2 /home/max/sources/ape/Source/Common/Types.h File Reference

```
#include <cstdint>
```

Typedefs

- using u8 = uint8_t
  A 8-bit unsigned integer (unsigned byte)
- using i8 = int8_t
  A 8-bit signed integer (unsigned byte)
- using u16 = uint16_t
  A 16-bit unsigned integer (unsigned word)
- using i16 = int16_t
  A 16-bit signed integer (unsigned word)
- using u32 = uint32_t
  A 32-bit unsigned integer.
- using i32 = int32_t
  A 32-bit signed integer.
- using u64 = uint64_t
  A 64-bit unsigned integer.
- using i64 = int64_t
  A 64-bit signed integer.

8.3 /home/max/sources/ape/Source/Core/CPU/CPU.h File Reference

```
#include <atomic>
#include <functional>
#include "Common/Logger.h"
#include "Common/String.h"
#include "Common/Types.h"
#include "Core/CPU/Exception.h"
#include "Core/CPU/Instruction.h"
#include "Core/Memory.h"
```

Classes

- union Core::CPU::GPR

Namespaces

- Core::CPU

  Representation of the Central Processing Unit.

Typedefs

- using **Core::CPU::StateCallbackFunc** = std::function< void(State)>

Enumerations

- enum **RepeatMode** : u8 { **None**, **Repeat**, **Repeat_Zero**, **Repeat_Non_Zero** }
- enum **State** : u8 { **Stopped**, **Running**, **Paused** }

Functions

- void Core::CPU::Tick ()

  Execute one CPU cycle.
- void Core::CPU::Start ()

  Execute instructions until shutdown is requested.
- void **Core::CPU::RegisterStateChangedCallback** (StateCallbackFunc fnc)
- void **Core::CPU::UnregisterStateChangedCallback** (StateCallbackFunc fnc)
- void Core::CPU::Stop ()

  Stop the CPU.
- void **Core::CPU::SetPaused** (bool value)
- bool **Core::CPU::IsRunning** ()
- bool **Core::CPU::IsPaused** ()
- State **Core::CPU::GetState** ()
- u16 **Core::CPU::PrefixToValue** (Instruction::SegmentPrefix prefix)
- PRIVATE void **Core::CPU::UpdateZF** (u16 value)
- void **Core::CPU::UpdatePF** (u16 value)
- void **Core::CPU::UpdateSF** (i16 value)
- template<typename T >
  void **Core::CPU::UpdateOF** (i32 value)
- template<typename T >
  void **Core::CPU::UpdateCF** (i32 value)
- void **Core::CPU::CallInterrupt** (u8 vector)
- bool **Core::CPU::CallBIOSInterrupt** (u8 vector)
- bool **Core::CPU::CallMSDOSInterrupt** (u8 vector)

8.4 /home/max/sources/ape/Source/Core/CPU/Exception.h File Reference

```
#include <exception>
#include "Core/CPU/Instruction.h"
```

Classes

- class Core::CPU::CPUException

  Base class thrown by every CPU related exception.
- class Core::CPU::InvalidInstructionException
- class Core::CPU::UnhandledInstructionException

  Thrown when the CPU hits an unimplemented Instruction::Type.
- class Core::CPU::UnhandledParameterException

  Thrown when the CPU hits an unimplemented Instruction::Parameter::Type.
- class Core::CPU::InvalidParameterException

  Thrown when the CPU hits an invalid mod byte that can't be decoded.
- class Core::CPU::ParameterLengthMismatchException
- class Core::CPU::UnsupportedParameterException

  Thrown when a instruction doesn't support the parameter type provided.
- class Core::CPU::UnhandledInterruptException

  Thrown when a interrupt is neither handled by software nor hardware.

Namespaces

- Core::CPU

  Representation of the Central Processing Unit.

8.5 /home/max/sources/ape/Source/Core/CPU/Instruction.h File Reference

```
#include <map>
#include <string>
#include <vector>
#include "Common/Types.h"
```

Classes

- class Core::CPU::Instruction

  High-Level representation of a instruction.
- class Core::CPU::Instruction::Parameter

  High-Level representation of a instruction parameter.

Namespaces

- Core::CPU

  Representation of the Central Processing Unit.

Functions

- std::string Core::CPU::TypeToString (const Instruction::Type &type)

  Get the corresponding nmoroic for the Type provided.
- std::string Core::CPU::ParameterTypeToString (const Instruction::Parameter::Type &type, Instruction::↩ SegmentPrefix prefix=Instruction::SegmentPrefix::None)

  Get a human readable form of the Parameter::Type provided.
- bool Core::CPU::ParameterNeedsResolving (const Instruction::Parameter::Type &parameter)

  Checks whether this Parameter::Type needs resolving.

8.6 /home/max/sources/ape/Source/Core/HW/FloppyDrive.h File Reference

```
#include <fstream>
#include <memory>
#include <string>
#include <vector>
#include "Common/Types.h"
```

Namespaces

- Core::HW::FloppyDrive

  Representation of a floppy drive.

Functions

- bool Core::HW::FloppyDrive::Insert (const std::string &path)

  Insert an image into the drive.
- bool Core::HW::FloppyDrive::HasDisc ()

  Check if a disc is present.
- u32 Core::HW::FloppyDrive::GetSize ()

  Get the size of the inserted disc.
- bool Core::HW::FloppyDrive::IsBootable ()

  Check if the provided image is bootable.
- u32 Core::HW::FloppyDrive::GetSectorSize ()

  Get size of a floppy.
- u32 Core::HW::FloppyDrive::GetSectorsPerTrack ()

  Get sectors per track.
- u32 Core::HW::FloppyDrive::GetHeadCount ()

  Get head count.
- void Core::HW::FloppyDrive::Eject ()

  Eject the image.
- bool Core::HW::FloppyDrive::Read (u32 offset, u32 size, u8 *buffer)

  Read data from the image.
- bool **Core::HW::FloppyDrive::Read** (u8 cylinder, u8 head, u8 sector, u8 count, u8 *buffer)
- bool **Core::HW::FloppyDrive::GuessFormat** ()

8.7 /home/max/sources/ape/Source/Core/Machine.h File Reference

```
#include <string>
```

Namespaces

- Core::Machine

  Representation of a PC.

Functions

- void **Core::Machine::Init** ()
- bool Core::Machine::BootFloppy ()

  Boot the machine from the floppy drive.
- void Core::Machine::Stop ()

  Stop the machine.
- void Core::Machine::Pause ()

  Pause the machine (Or unpause it if it's paused already)
- bool Core::Machine::BootCOM (const std::string &file, const std::string &&parameters="")

  Directly execute a COM file.

8.8 /home/max/sources/ape/Source/Core/Memory.h File Reference

```
#include <vector>
#include "Common/Types.h"
```

Namespaces

- Core::Memory

  Wrapper around emulated RAM.

Functions

- std::vector< u8 > & Core::Memory::Get ()

  Get the contents of RAM.
- u32 Core::Memory::VirtToPhys (u16 segment, u16 offset)

  Converts a virtual address to an absolute one.
- template<typename T >
  T & **Core::Memory::Get** (u16 segment, u16 offset)
- template<typename T >
  T * **Core::Memory::GetPtr** (u16 segment, u16 offset)

Index