Andrew Todd 4380 VM v.1

Andrew Todd | October 4, 2023

Instructions

ADD

DIV

JMP_L

LDB_L

LDR_L

MOV

MUL

SUB

TRP

VMFramework

{ PointerMath is a static class }

PointerMath

+ PointerMath() {delete}
+ ~PointerMath() {delete}
+ PointerMath(const PointerMath&) {delete}
+ operator=(const PointerMath&) {delete}
+ PointerMath(PointerMath&&) {delete}
+ operator=(PointerMath&&) {delete}

+ AlignForward(void*, const uint8_t&): void* {static, inline}
+ AlignForward(const void*, const uint8_t&): const void* {static, inline}
+ AlignBackward(void*, const uint8_t&): void* {static, inline}
+ AlignBackward(const void*, const uint8_t&): const void* {static, inline}
+ AlignForwardAdjustment(const void*, const uint8_t&): uint8_t {static, inline}
+ AlignForwardAdjustmentWithHeader(const void*, const uint8_t&, const uint8_t&): uint8_t {static, inline}
+ AlignBackwardAdjustment(const void*, const uint8_t&): uint8_t {static, inline}
+ Add(void*, const size_t&): void* {static, inline}
+ Add(const void*, const size_t&): const void* {static, inline}
+ Subtract(void*, const size_t&): void* {static, inline}
+ Subtract(const void*, const size_t&): const void* {static, inline}

Uses

Allocator

m_start: void*
m_usedMemory: size_t
m_numOfAllocations: size_t

+ m_size: size_t

Allocator(const size_t&, void*)
~Allocator() {virtual}

+ Allocate(const size_t&, const uint8_t&): void* {virtual}
+ Deallocate(void*): void {virtual}
+ ClearAllocator(): void

+ GetStart() const: void*
+ GetSize() const: size_t
+ GetUsedMemory() const: size_t
+ GetNumAllocations() const: size_t

**PointerMath class is contained in Allocator.h

ISA: InstructionSetArchitecture

GPRegisterType: typename
RegisterType: IntExpr
ProcessType:Process

ISA

cexp_setSize: cosntexpr size_t
m_instructionSet: const Instruction*[]

+ ISA(Instruction*[], const size_t&)
+ ~ISA()

+ ISA(const ISA&) {delete}
+ operator=(const ISA&) {delete}
+ ISA(ISA&&) {delete}
+ operator=(ISA&&) {delete}

+ operator[]: Instruction*

_setSize deturmines the length of the _instructionSet array, not all indexes have to be used

For _instructionSet: index num coresponds to Instruction OpCode

operator[] returns function pointer to Instruction's Op

GPRegisterType: typename
RegisterType: IntExpr
ProcessType: Process

<<abstract>>
Instruction

+ opcode: RegisterType

+ Instruction(const RegisterType&)

+ Instruction(const Instruction&) {delete}
+ operator=(const Instruction&) {delete}
+ Instruction(Instruction&&) {delete}
+ operator=(Instruction&&) {delete}

+ Op(ProcessType*): void {pure virtual}

All structs in the Instructions package inherit from *Instruction*

{ Instruction is friend of Machine }

{ Instruction is friend of Process }

Machine uses injected-class-name to create Process and Registers

Derived: typename
GPRegisterType: typename
RegisterType: IntExpr
ProcessType: Process
ISA: ISA

Machine

s_instance: Derived* {static}

m_processes: Vector<ProcessType*>
M_processThreads: Vector<std::thread>

m_memoryManager: MemoryManager

m_ISA: ISA

m_programSegment: uint8_t*
m_codeSegment: uint8_t*

m_programSize: size_t

sharedMutex: std::shared_mutex

Machine()
~Machine()

+ Machine(const Machine&) {delete}
+ operator=(const Machine&) {delete}
+ Machine(Machine&&) {delete}
+ operator(Machine&&) {delete}

LoadProgram(char*): void {virtual}
SpawnProcess(void*): void {virtual}

+ GetInstance(): Derived* {static}
+ StartUp(): void
+ StartUp(char*): void
+ WaitForProcess(): void
+ ShutDown(): void
+ LaunchProgram(char*): void {virtual}
# LaunchProgram_INTERNAL(char*): void {virtual}

Derived: typename
GPRegisterType: typename
RegisterType: IntExpr

Process

m_registers: Registers
m_stack: StackAllocator*

_ISA: const ISA*

programSegment: uint8_t*
codeSegment: uint8_t*

_machineMutex: const shared_mutex&

_run: bool

Fetch: void {virtual}{inline}
Increment: void {virtual}{inline}
Decode: void {virtual}{inline}
Execute: void {virtual}{inline}

+ Process(void*, StackAllocator*, uint8_t*, uint8_t*, shared_mutex&, ISA*)
# ~Process()

+ Run(): void
+ Stop(): void

GPRegisterType: typedef
RegisterType: IntExpr

Registers

+ gpRegisters: GPRegisterType[]
+ PC: RegisterType

Registers is defined in Process.h

MemoryManager

- m_applicationMemory: void*
- s_instance: MemoryManager* {static}

+ m_systemAllocator: StackAllocator*

- MemoryManager()
- ~MemoryManager()

+ GetInstance(): MemoryManager* {static}

+ StartUp(): void
+ ShutDown(): void

+ MemoryManager(const MemoryManager&) {delete}
+ operator=(const MemoryManager&) {delete}
+ MemoryManager(MemoryManager&&) {delete}
+ operator=(MemoryManager&&) {delete}

StackAllocator

- m_prev_position: void* {_DEBUG}
- m_current_position: void*

+ StackAllocator(const size_t&, void*)
+ ~StackAllocator()

+ Allocate(const size_t&, const uint8_t&) override: void*
+ Deallocate(void*) override: void

+ StackAllocator(const StackAllocator&) {delete}
+ operator=(const StackAllocator&) {delete}

AllocationHeader

+ m_prev_address: void* {_DEBUG}
+ m_adjustment: uint8_t

VM_4380

Process_4380

opcode: int32_t
operandOne: int32_t
operandTwo: int32_t
_instruction: Instruction*

{ All the Instructions are declaired as friends of Process_4380 }

VM4380

ISA_4380

{ ISA_4380 contains an array of all the Instructions }