///----------------------------------------------------------------------------

///

/// InterpreterStackFrame::Process

///

/// Process() processes a single loop of execution for the current

/// JavascriptFunction being executed:

/// - Individual instructions are dispatched to specific handlers for different

/// OpCodes.

///

///----------------------------------------------------------------------------

#if ENABLE\_PROFILE\_INFO

#define INTERPRETERLOOPNAME ProcessProfiled

#define PROVIDE\_INTERPRETERPROFILE

#include "Interpreterloop.inl"

#undef PROVIDE\_INTERPRETERPROFILE

#undef INTERPRETERLOOPNAME

#endif

#define INTERPRETERLOOPNAME ProcessUnprofiled

#include "Interpreterloop.inl"

#undef INTERPRETERLOOPNAME

#ifndef TEMP\_DISABLE\_ASMJS

#define INTERPRETERLOOPNAME ProcessAsmJs

#define INTERPRETER\_ASMJS

#include "InterpreterProcessOpCodeAsmJs.h"

#include "Interpreterloop.inl"

#undef INTERPRETER\_ASMJS

#undef INTERPRETERLOOPNAME

#endif

// For now, always collect profile data when debugging,

// otherwise the backend will be confused if there's no profile data.

#define INTERPRETERLOOPNAME ProcessWithDebugging

#define PROVIDE\_DEBUGGING

#if ENABLE\_PROFILE\_INFO

#define PROVIDE\_INTERPRETERPROFILE

#endif

#include "Interpreterloop.inl"

#if ENABLE\_PROFILE\_INFO

#undef PROVIDE\_INTERPRETERPROFILE

#endif

#undef PROVIDE\_DEBUGGING

#undef INTERPRETERLOOPNAME

Var InterpreterStackFrame::Process()

{

#if ENABLE\_PROFILE\_INFO

class AutoRestore

{

private:

InterpreterStackFrame \*const interpreterStackFrame;

const uint32 savedSwitchProfileModeOnLoopEndNumber;

const bool savedIsAutoProfiling;

const bool savedSwitchProfileMode;

public:

AutoRestore(InterpreterStackFrame \*const interpreterStackFrame)

: interpreterStackFrame(interpreterStackFrame),

savedIsAutoProfiling(interpreterStackFrame->isAutoProfiling),

savedSwitchProfileMode(interpreterStackFrame->switchProfileMode),

savedSwitchProfileModeOnLoopEndNumber(interpreterStackFrame->switchProfileModeOnLoopEndNumber)

{

}

~AutoRestore()

{

interpreterStackFrame->isAutoProfiling = savedIsAutoProfiling;

interpreterStackFrame->switchProfileMode = savedSwitchProfileMode;

interpreterStackFrame->switchProfileModeOnLoopEndNumber = savedSwitchProfileModeOnLoopEndNumber;

}

} autoRestore(this);

#endif

if ((m\_flags & Js::InterpreterStackFrameFlags\_FromBailOut) && !(m\_flags & InterpreterStackFrameFlags\_ProcessingBailOutFromEHCode))

{

if (this->ehBailoutData)

{

m\_flags |= Js::InterpreterStackFrameFlags\_ProcessingBailOutFromEHCode;

EHBailoutData \* topLevelEHBailoutData = this->ehBailoutData;

while (topLevelEHBailoutData->parent->nestingDepth != -1)

{

topLevelEHBailoutData->parent->child = topLevelEHBailoutData;

topLevelEHBailoutData = topLevelEHBailoutData->parent;

}

ProcessTryCatchBailout(topLevelEHBailoutData, this->ehBailoutData->nestingDepth);

m\_flags &= ~Js::InterpreterStackFrameFlags\_ProcessingBailOutFromEHCode;

this->ehBailoutData = nullptr;

}

}

#ifndef TEMP\_DISABLE\_ASMJS

FunctionBody \*const functionBody = GetFunctionBody();

if( functionBody->GetIsAsmjsMode() )

{

AsmJsFunctionInfo\* asmInfo = functionBody->GetAsmJsFunctionInfo();

if (asmInfo)

{

AlignMemoryForAsmJs();

Var returnVar = ProcessAsmJs();

#if DBG\_DUMP

if( PHASE\_TRACE( AsmjsFunctionEntryPhase, functionBody ) )

{

--AsmJsCallDepth;

if( AsmJsCallDepth )

{

Output::Print( L"%\*c}", AsmJsCallDepth, ' ' );

}

else

{

Output::Print( L"}" );

}

switch( asmInfo->GetReturnType().which() )

{

case AsmJsRetType::Void:

break;

case AsmJsRetType::Signed:

Output::Print( L" = %d", JavascriptMath::ToInt32( returnVar, scriptContext ) );

break;

case AsmJsRetType::Float:

case AsmJsRetType::Double:

Output::Print( L" = %.4f", JavascriptConversion::ToNumber( returnVar, scriptContext ) );

break;

default:

break;

}

Output::Print( L";\n" );

}

#endif

return returnVar;

}

else

{

Assert(functionBody->GetAsmJsModuleInfo());

return ProcessAsmJsModule();

}

}

#endif

#if ENABLE\_PROFILE\_INFO

switchProfileMode = false;

switchProfileModeOnLoopEndNumber = 0u - 1;

#endif

ByteCodeDumper::Dump(functionBody);

#if ENABLE\_PROFILE\_INFO

const ExecutionMode interpreterExecutionMode =

functionBody->GetInterpreterExecutionMode(!!(GetFlags() & InterpreterStackFrameFlags\_FromBailOut));

if(interpreterExecutionMode == ExecutionMode::ProfilingInterpreter)

{

isAutoProfiling = false;

return ProcessProfiled();

}

Assert(

interpreterExecutionMode == ExecutionMode::Interpreter ||

interpreterExecutionMode == ExecutionMode::AutoProfilingInterpreter);

isAutoProfiling = interpreterExecutionMode == ExecutionMode::AutoProfilingInterpreter;

Var result;

while(true)

{

Assert(!switchProfileMode);

result = ProcessUnprofiled();

Assert(!(switchProfileMode && result));

if(switchProfileMode)

{

switchProfileMode = false;

}

else

{

break;

}

Assert(isAutoProfiling);

#if DBG\_DUMP

if(PHASE\_TRACE(InterpreterAutoProfilePhase, functionBody))

{

wchar\_t debugStringBuffer[MAX\_FUNCTION\_BODY\_DEBUG\_STRING\_SIZE];

Output::Print(L"InterpreterAutoProfile - Func %s - Started profiling\n", functionBody->GetDebugNumberSet(debugStringBuffer));

Output::Flush();

}

#endif

Assert(!switchProfileMode);

result = ProcessProfiled();

Assert(!(switchProfileMode && result));

if(switchProfileMode)

{

switchProfileMode = false;

}

else

{

break;

}

Assert(isAutoProfiling);

#if DBG\_DUMP

if(PHASE\_TRACE(InterpreterAutoProfilePhase, functionBody))

{

wchar\_t debugStringBuffer[MAX\_FUNCTION\_BODY\_DEBUG\_STRING\_SIZE];

Output::Print(L"InterpreterAutoProfile - Func %s - Stopped profiling\n", functionBody->GetDebugNumberSet(debugStringBuffer));

Output::Flush();

}

#endif

}

return result;

#else

return ProcessUnprofiled();

#endif

}

//-------------------------------------------------------------------------------------------------------

// Copyright (C) Microsoft. All rights reserved.

// Licensed under the MIT license. See LICENSE.txt file in the project root for full license information.

//-------------------------------------------------------------------------------------------------------

// Shared intepreter loop

//

// This holds the single definition of the interpreter loop.

// It allows for configurable copies of the loop that do extra work without

// impacting the mainline performance. (for example the debug loop can simply

// check a bit without concern for impacting the nondebug mode.)

#if defined(INTERPRETER\_ASMJS) && !defined(TEMP\_DISABLE\_ASMJS)

#define INTERPRETER\_OPCODE OpCodeAsmJs

#else

#define INTERPRETER\_OPCODE OpCode

#endif

#ifdef PROVIDE\_DEBUGGING

#define DEBUGGING\_LOOP 1

#else

#define DEBUGGING\_LOOP 0

#endif

#ifdef PROVIDE\_INTERPRETERPROFILE

#define INTERPRETERPROFILE 1

#define PROFILEDOP(prof, unprof) prof

#else

#define INTERPRETERPROFILE 0

#define PROFILEDOP(prof, unprof) unprof

#endif

#if defined (DBG)

// Win8 516184: Huge switch with lots of labels each having a few locals on ARM.DBG causes each occurrence

// of this function (call of a javascript function in interpreter mode) to take 7+KB stack space

// (without optimizations the compiler accounts for ALL locals inside case labels when allocating space on stack

// for locals - SP does not change inside the function). On other platforms this is still huge but better than ARM.

// So, for DBG turn on optimizations to prevent this huge loss of stack.

#pragma optimize("g", on)

#endif

Var Js::InterpreterStackFrame::INTERPRETERLOOPNAME()

{

PROBE\_STACK(scriptContext, Js::Constants::MinStackInterpreter);

if (!this->closureInitDone)

{

// If this is the start of the function, then we've waited until after the stack probe above

// to set up the FD/SS pointers, so do it now.

Assert(this->m\_reader.GetCurrentOffset() == 0);

this->InitializeClosures();

}

Assert(this->returnAddress != nullptr);

AssertMsg(m\_arguments == NULL || Js::ArgumentsObject::Is(m\_arguments), "Bad arguments!");

// IP Passing in the interpreter:

// We keep a local copy of the bytecode's instruction pointer and

// pass it by reference to the bytecode reader.

// This allows the optimizer to recognize that the local (held in a register)

// dominates the copy in the reader.

// The effect is our dispatch loop is significantly smaller in the common case

// on optimized builds.

//

// For checked builds this does mean we are incrementing 2 different counters to

// track the ip.

const byte\* ip = m\_reader.GetIP();

while (true)

{

INTERPRETER\_OPCODE op = ReadByteOp<INTERPRETER\_OPCODE>(ip);

#ifdef ENABLE\_BASIC\_TELEMETRY

if( TELEMETRY\_OPCODE\_OFFSET\_ENABLED )

{

OpcodeTelemetry& opcodeTelemetry = this->scriptContext->GetTelemetry().GetOpcodeTelemetry();

opcodeTelemetry.ProgramLocationFunctionId ( this->function->GetFunctionInfo()->GetLocalFunctionId() );

opcodeTelemetry.ProgramLocationBytecodeOffset( this->m\_reader.GetCurrentOffset() );

}

#endif

#if DEBUGGING\_LOOP

if (this->scriptContext->GetThreadContext()->GetDebugManager()->stepController.IsActive() &&

this->scriptContext->GetThreadContext()->GetDebugManager()->stepController.IsStepComplete\_AllowingFalsePositives(this))

{

// BrLong is used for branch island, we don't want to break over there, as they don't belong to any statement. Just skip this.

if (!InterpreterStackFrame::IsBrLong(op, ip) && !this->m\_functionBody->GetUtf8SourceInfo()->GetIsLibraryCode())

{

uint prevOffset = m\_reader.GetCurrentOffset();

InterpreterHaltState haltState(STOP\_STEPCOMPLETE, m\_functionBody);

this->scriptContext->GetDebugContext()->GetProbeContainer()->DispatchStepHandler(&haltState, &op);

if (prevOffset != m\_reader.GetCurrentOffset())

{

// The location of the statement has been changed, setnextstatement was called.

// Reset m\_outParams and m\_outSp as before SetNext was called, we could be in the middle of StartCall.

// It's fine to do because SetNext can only be done to a statement -- function-level destination,

// and can't land to an expression inside call.

ResetOut();

ip = m\_reader.GetIP();

continue;

}

}

}

// The break opcode will be handled later in the switch block.

if (op != OpCode::Break && this->scriptContext->GetThreadContext()->GetDebugManager()->asyncBreakController.IsBreak())

{

if (!InterpreterStackFrame::IsBrLong(op, ip) && !this->m\_functionBody->GetUtf8SourceInfo()->GetIsLibraryCode())

{

uint prevOffset = m\_reader.GetCurrentOffset();

InterpreterHaltState haltState(STOP\_ASYNCBREAK, m\_functionBody);

this->scriptContext->GetDebugContext()->GetProbeContainer()->DispatchAsyncBreak(&haltState);

if (prevOffset != m\_reader.GetCurrentOffset())

{

// The location of the statement has been changed, setnextstatement was called.

ip = m\_reader.GetIP();

continue;

}

}

}

SWAP\_BP\_FOR\_OPCODE:

#endif

switch (op)

{

case INTERPRETER\_OPCODE::Ret:

{

//

// Return "Reg: 0" as the return-value.

// - JavaScript functions always return a value, and this value is always

// accessible to the caller. For an empty "return;" or exiting the end of the

// function's body, it is assumed that the byte-code author

// (ByteCodeGenerator) will load 'undefined' into R0.

// - If R0 has not explicitly been set, it will contain whatever garbage value

// was last set.

//

this->retOffset = m\_reader.GetCurrentOffset();

m\_reader.Empty(ip);

return GetReg((RegSlot)0);

}

case INTERPRETER\_OPCODE::Yield:

{

m\_reader.Reg2\_Small(ip);

return GetReg(GetFunctionBody()->GetYieldRegister());

}

#define DEF2(x, op, func) PROCESS\_##x(op, func)

#define DEF3(x, op, func, y) PROCESS\_##x(op, func, y)

#define DEF2\_WMS(x, op, func) PROCESS\_##x##\_COMMON(op, func, \_Small)

#define DEF3\_WMS(x, op, func, y) PROCESS\_##x##\_COMMON(op, func, y, \_Small)

#define DEF4\_WMS(x, op, func, y, t) PROCESS\_##x##\_COMMON(op, func, y, \_Small, t)

#include "InterpreterHandler.inl"

case INTERPRETER\_OPCODE::Leave:

// Return the continuation address to the helper.

// This tells the helper that control left the scope without completing the try/handler,

// which is particularly significant when executing a finally.

m\_reader.Empty(ip);

return (Var)this->m\_reader.GetCurrentOffset();

case INTERPRETER\_OPCODE::LeaveNull:

// Return to the helper without specifying a continuation address,

// indicating that the handler completed without jumping, so exception processing

// should continue.

m\_reader.Empty(ip);

return nullptr;

case INTERPRETER\_OPCODE::ExtendedOpcodePrefix:

{

ip = [this](const byte \* ip) -> const byte \*

{

INTERPRETER\_OPCODE op = (INTERPRETER\_OPCODE)(ReadByteOp<INTERPRETER\_OPCODE>(ip

#if DBG\_DUMP

, true

#endif

) + (INTERPRETER\_OPCODE::ExtendedOpcodePrefix << 8));

switch (op)

{

#define EXDEF2(x, op, func) PROCESS\_##x(op, func)

#define EXDEF3(x, op, func, y) PROCESS\_##x(op, func, y)

#define EXDEF2\_WMS(x, op, func) PROCESS\_##x##\_COMMON(op, func, \_Small)

#define EXDEF3\_WMS(x, op, func, y) PROCESS\_##x##\_COMMON(op, func, y, \_Small)

#define EXDEF4\_WMS(x, op, func, y, t) PROCESS\_##x##\_COMMON(op, func, y, \_Small, t)

#include "InterpreterHandler.inl"

default:

// Help the C++ optimizer by declaring that the cases we

// have above are sufficient

AssertMsg(false, "dispatch to bad opcode");

\_\_assume(false);

};

return ip;

}(ip);

#if ENABLE\_PROFILE\_INFO

if (switchProfileMode)

{

// Aborting the current interpreter loop to switch the profile mode

return nullptr;

}

#endif

break;

}

case INTERPRETER\_OPCODE::MediumLayoutPrefix:

{

Var yieldValue = nullptr;

ip = [this, &yieldValue](const byte \* ip) -> const byte \*

{

INTERPRETER\_OPCODE op = ReadByteOp<INTERPRETER\_OPCODE>(ip);

switch (op)

{

case INTERPRETER\_OPCODE::Yield:

m\_reader.Reg2\_Medium(ip);

yieldValue = GetReg(GetFunctionBody()->GetYieldRegister());

break;

#define DEF2\_WMS(x, op, func) PROCESS\_##x##\_COMMON(op, func, \_Medium)

#define DEF3\_WMS(x, op, func, y) PROCESS\_##x##\_COMMON(op, func, y, \_Medium)

#define DEF4\_WMS(x, op, func, y, t) PROCESS\_##x##\_COMMON(op, func, y, \_Medium, t)

#include "InterpreterHandler.inl"

default:

// Help the C++ optimizer by declaring that the cases we

// have above are sufficient

AssertMsg(false, "dispatch to bad opcode");

\_\_assume(false);

}

return ip;

}(ip);

if (yieldValue != nullptr)

{

return yieldValue;

}

#if ENABLE\_PROFILE\_INFO

if (switchProfileMode)

{

// Aborting the current interpreter loop to switch the profile mode

return nullptr;

}

#endif

break;

}

case INTERPRETER\_OPCODE::ExtendedMediumLayoutPrefix:

{

#ifndef INTERPRETER\_ASMJS // Asmjs doesn't have any extended opcodes for now, remove that case

ip = [this](const byte \* ip) -> const byte \*

{

INTERPRETER\_OPCODE op = (INTERPRETER\_OPCODE)(ReadByteOp<INTERPRETER\_OPCODE>(ip

#if DBG\_DUMP

, true

#endif

) + (INTERPRETER\_OPCODE::ExtendedOpcodePrefix << 8));

switch (op)

{

#define EXDEF2\_WMS(x, op, func) PROCESS\_##x##\_COMMON(op, func, \_Medium)

#define EXDEF3\_WMS(x, op, func, y) PROCESS\_##x##\_COMMON(op, func, y, \_Medium)

#define EXDEF4\_WMS(x, op, func, y, t) PROCESS\_##x##\_COMMON(op, func, y, \_Medium, t)

#include "InterpreterHandler.inl"

default:

// Help the C++ optimizer by declaring that the cases we

// have above are sufficient

AssertMsg(false, "dispatch to bad opcode");

\_\_assume(false);

};

return ip;

}(ip);

#if ENABLE\_PROFILE\_INFO

if (switchProfileMode)

{

// Aborting the current interpreter loop to switch the profile mode

return nullptr;

}

#endif

#endif

break;

}

case INTERPRETER\_OPCODE::LargeLayoutPrefix:

{

Var yieldValue = nullptr;

ip = [this, &yieldValue](const byte \* ip) -> const byte \*

{

INTERPRETER\_OPCODE op = ReadByteOp<INTERPRETER\_OPCODE>(ip);

switch (op)

{

case INTERPRETER\_OPCODE::Yield:

m\_reader.Reg2\_Large(ip);

yieldValue = GetReg(GetFunctionBody()->GetYieldRegister());

break;

#define DEF2\_WMS(x, op, func) PROCESS\_##x##\_COMMON(op, func, \_Large)

#define DEF3\_WMS(x, op, func, y) PROCESS\_##x##\_COMMON(op, func, y, \_Large)

#define DEF4\_WMS(x, op, func, y, t) PROCESS\_##x##\_COMMON(op, func, y, \_Large, t)

#include "InterpreterHandler.inl"

default:

// Help the C++ optimizer by declaring that the cases we

// have above are sufficient

AssertMsg(false, "dispatch to bad opcode");

\_\_assume(false);

}

return ip;

}(ip);

if (yieldValue != nullptr)

{

return yieldValue;

}

#if ENABLE\_PROFILE\_INFO

if(switchProfileMode)

{

// Aborting the current interpreter loop to switch the profile mode

return nullptr;

}

#endif

break;

}

case INTERPRETER\_OPCODE::ExtendedLargeLayoutPrefix:

{

#ifndef INTERPRETER\_ASMJS // Asmjs doesn't have any extended opcodes for now, remove that case

ip = [this](const byte \* ip) -> const byte \*

{

INTERPRETER\_OPCODE op = (INTERPRETER\_OPCODE)(ReadByteOp<INTERPRETER\_OPCODE>(ip

#if DBG\_DUMP

, true

#endif

) + (INTERPRETER\_OPCODE::ExtendedOpcodePrefix << 8));

switch (op)

{

#define EXDEF2\_WMS(x, op, func) PROCESS\_##x##\_COMMON(op, func, \_Large)

#define EXDEF3\_WMS(x, op, func, y) PROCESS\_##x##\_COMMON(op, func, y, \_Large)

#define EXDEF4\_WMS(x, op, func, y, t) PROCESS\_##x##\_COMMON(op, func, y, \_Large, t)

#include "InterpreterHandler.inl"

default:

// Help the C++ optimizer by declaring that the cases we

// have above are sufficient

AssertMsg(false, "dispatch to bad opcode");

\_\_assume(false);

};

return ip;

}(ip);

#if ENABLE\_PROFILE\_INFO

if(switchProfileMode)

{

// Aborting the current interpreter loop to switch the profile mode

return nullptr;

}

#endif

#endif

break;

}

case INTERPRETER\_OPCODE::EndOfBlock:

{

// Note that at this time though ip was advanced by 'OpCode op = ReadByteOp<INTERPRETER\_OPCODE>(ip)',

// we haven't advanced m\_reader.m\_currentLocation yet, thus m\_reader.m\_currentLocation still points to EndOfBLock,

// and that +1 will point to 1st byte past the buffer.

Assert(m\_reader.GetCurrentOffset() + sizeof(byte) == m\_functionBody->GetByteCode()->GetLength());

//

// Reached an "OpCode::EndOfBlock" so need to exit this interpreter loop because

// there is no more byte-code to execute.

// - This prevents us from accessing random memory as byte-codes.

// - Functions should contain an "OpCode::Ret" instruction to organize an

// orderly return.

//

#if DEBUGGING\_LOOP

// However, during debugging an exception can be skipped which causes the

// statement that caused to exception to be skipped. If this statement is

// the statement that contains the OpCode::Ret then the EndOfBlock will

// be executed. In these cases it is sufficient to return undefined.

return this->scriptContext->GetLibrary()->GetUndefined();

#else

return nullptr;

#endif

}

case INTERPRETER\_OPCODE::Break:

{

#if DEBUGGING\_LOOP

// The reader has already advanced the IP:

if (this->m\_functionBody->ProbeAtOffset(m\_reader.GetCurrentOffset(), &op))

{

uint prevOffset = m\_reader.GetCurrentOffset();

InterpreterHaltState haltState(STOP\_BREAKPOINT, m\_functionBody);

this->scriptContext->GetDebugContext()->GetProbeContainer()->DispatchProbeHandlers(&haltState);

if (prevOffset != m\_reader.GetCurrentOffset())

{

// The location of the statement has been changed, setnextstatement was called.

ip = m\_reader.GetIP();

continue;

}

// Jump back to the start of the switch.

goto SWAP\_BP\_FOR\_OPCODE;

}

else

{

#if DEBUGGING\_LOOP

// an inline break statement rather than a probe

if (!this->scriptContext->GetThreadContext()->GetDebugManager()->stepController.ContinueFromInlineBreakpoint())

{

uint prevOffset = m\_reader.GetCurrentOffset();

InterpreterHaltState haltState(STOP\_INLINEBREAKPOINT, m\_functionBody);

this->scriptContext->GetDebugContext()->GetProbeContainer()->DispatchInlineBreakpoint(&haltState);

if (prevOffset != m\_reader.GetCurrentOffset())

{

// The location of the statement has been changed, setnextstatement was called.

ip = m\_reader.GetIP();

continue;

}

}

#endif

// Consume after dispatching

m\_reader.Empty(ip);

}

#else

m\_reader.Empty(ip);

#endif

break;

}

default:

// Help the C++ optimizer by declaring that the cases we

// have above are sufficient

AssertMsg(false, "dispatch to bad opcode");

\_\_assume(false);

}

}

}

#if defined (DBG)

// Restore optimizations to what's specified by the /O switch.

#pragma optimize("", on)

#endif

#undef DEBUGGING\_LOOP

#undef INTERPRETERPROFILE

#undef PROFILEDOP

#undef INTERPRETER\_OPCODE

//-------------------------------------------------------------------------------------------------------

// Copyright (C) Microsoft. All rights reserved.

// Licensed under the MIT license. See LICENSE.txt file in the project root for full license information.

//-------------------------------------------------------------------------------------------------------

// Default all macro to nothing

#ifndef DEF2

#define DEF2(process, op, func)

#endif

#ifndef DEF3

#define DEF3(process, op, func, y)

#endif

#ifndef DEF2\_WMS

#define DEF2\_WMS(process, op, func)

#endif

#ifndef DEF3\_WMS

#define DEF3\_WMS(process, op, func, y)

#endif

#ifndef DEF4\_WMS

#define DEF4\_WMS(process, op, func, y, t)

#endif

#ifndef EXDEF2

#define EXDEF2(process, op, func)

#endif

#ifndef EXDEF3

#define EXDEF3(process, op, func, y)

#endif

#ifndef EXDEF2\_WMS

#define EXDEF2\_WMS(process, op, func)

#endif

#ifndef EXDEF3\_WMS

#define EXDEF3\_WMS(process, op, func, y)

#endif

#ifndef EXDEF4\_WMS

#define EXDEF4\_WMS(process, op, func, y, t)

#endif

#if defined(INTERPRETER\_ASMJS) && !defined(TEMP\_DISABLE\_ASMJS)

#include "InterpreterHandlerAsmJs.inl"

#else

DEF2 (FALLTHROUGH, EndSwitch, /\* Common case with Br \*/)

DEF2 (BR, Br, OP\_Br)

#ifdef BYTECODE\_BRANCH\_ISLAND

EXDEF2 (BRLONG, BrLong, OP\_Br)

#endif

DEF3 (CUSTOM, StartCall, OP\_StartCall, StartCall)

DEF2 (NOP, Nop, Empty)

DEF2\_WMS(NOP, Unused, Reg1)

DEF2\_WMS(IP\_TARG, ProfiledLoopStart, OP\_ProfiledLoopStart)

DEF2\_WMS(FALLTHROUGH, LoopBodyStart, /\* Common case with ProfiledLoopBodyStart \*/)

DEF2\_WMS(IP\_TARG, ProfiledLoopBodyStart, OP\_ProfiledLoopBodyStart)

DEF2\_WMS(IP\_TARG, ProfiledLoopEnd, OP\_ProfiledLoopEnd)

DEF2\_WMS(BRCMem, BrEq\_A, JavascriptOperators::Equal)

DEF2\_WMS(BRCMem, BrGt\_A, JavascriptOperators::Greater)

DEF2\_WMS(BRCMem, BrGe\_A, JavascriptOperators::GreaterEqual)

DEF2\_WMS(BRCMem, BrLt\_A, JavascriptOperators::Less)

DEF2\_WMS(BRCMem, BrLe\_A, JavascriptOperators::LessEqual)

DEF2\_WMS(BRCMem, BrNeq\_A, JavascriptOperators::NotEqual)

DEF2\_WMS(BRBMem\_ALLOW\_STACK, BrFalse\_A, OP\_BrFalse\_A)

DEF2\_WMS(BRBMem\_ALLOW\_STACK, BrTrue\_A, OP\_BrTrue\_A)

DEF2\_WMS(BRB\_ALLOW\_STACK, BrOnObject\_A, JavascriptOperators::IsObject)

DEF2\_WMS(BRB, BrNotNull\_A, OP\_BrNotNull\_A)

//Not emitted for byte code, keep it here for completeness

//EXDEF2\_WMS(BRB, BrUndecl\_A, OP\_BrUndecl\_A)

EXDEF2\_WMS(BRB, BrNotUndecl\_A, OP\_BrNotUndecl\_A)

DEF2\_WMS(FALLTHROUGH, Case, /\* Common case with BrSrEq\_A \*/)

DEF2\_WMS(BRCMem, BrSrEq\_A, JavascriptOperators::StrictEqual)

DEF2\_WMS(BRCMem, BrSrNeq\_A, JavascriptOperators::NotStrictEqual)

//Not emitted for byte code, keep it here for completeness

//DEF2 (BRS, BrHasSideEffects, JavascriptOperators::OP\_BrHasSideEffects)

DEF2 (BRS, BrNotHasSideEffects, JavascriptOperators::OP\_BrNotHasSideEffects)

EXDEF2 (BRPROP, BrOnHasProperty, OP\_BrOnHasProperty)

DEF2 (BRPROP, BrOnNoProperty, OP\_BrOnNoProperty)

DEF2 (BRLOCALPROP, BrOnNoLocalProperty, OP\_BrOnNoProperty)

DEF2 (BRENVPROP, BrOnNoEnvProperty, OP\_BrOnNoEnvProperty)

DEF2\_WMS(BRBS, BrFncNeqApply, JavascriptOperators::OP\_BrFncNeqApply)

//Not emitted for byte code, keep it here for completeness

//DEF2\_WMS(BRBS, BrFncEqApply, JavascriptOperators::OP\_BrFncEqApply)

DEF3\_WMS(CALL\_FLAGS\_None, CallI, OP\_CallI, CallI)

DEF3\_WMS(CALL\_FLAGS\_None, CallIExtended, OP\_CallIExtended, CallIExtended)

DEF3\_WMS(CALL\_FLAGS\_None, CallIExtendedFlags, OP\_CallIExtendedFlags, CallIExtendedFlags)

DEF3\_WMS(CALL\_FLAGS\_None, CallIFlags, OP\_CallIFlags, CallIFlags)

DEF3\_WMS(CALL\_FLAGS\_None, ProfiledCallI, PROFILEDOP(OP\_ProfiledCallI, OP\_CallI), ProfiledCallI)

DEF3\_WMS(CALL\_FLAGS\_None, ProfiledCallIExtended, PROFILEDOP(OP\_ProfiledCallIExtended, OP\_CallIExtended), ProfiledCallIExtended)

DEF3\_WMS(CALL\_FLAGS\_None, ProfiledCallIExtendedFlags, PROFILEDOP(OP\_ProfiledCallIExtendedFlags, OP\_CallIExtendedFlags), ProfiledCallIExtendedFlags)

DEF3\_WMS(CALL\_FLAGS\_None, ProfiledCallIWithICIndex, PROFILEDOP(OP\_ProfiledCallIWithICIndex, OP\_CallI), ProfiledCallIWithICIndex)

DEF3\_WMS(CALL\_FLAGS\_Value, ProfiledCallIExtendedWithICIndex, PROFILEDOP(OP\_ProfiledCallIExtendedWithICIndex, OP\_CallIExtended), ProfiledCallIExtendedWithICIndex)

DEF3\_WMS(CALL\_FLAGS\_None, ProfiledCallIExtendedFlagsWithICIndex, PROFILEDOP(OP\_ProfiledCallIExtendedFlagsWithICIndex, OP\_CallIExtendedFlags), ProfiledCallIExtendedFlagsWithICIndex)

DEF3\_WMS(CALL\_FLAGS\_None, ProfiledCallIFlagsWithICIndex, PROFILEDOP(OP\_ProfiledCallIFlags, OP\_CallIFlags), ProfiledCallIFlagsWithICIndex)

DEF3\_WMS(CALL\_FLAGS\_None, ProfiledCallIFlags, PROFILEDOP(OP\_ProfiledCallIFlags, OP\_CallIFlags), ProfiledCallIFlags)

DEF3\_WMS(CALL\_FLAGS\_None, ProfiledReturnTypeCallI, PROFILEDOP(OP\_ProfiledReturnTypeCallI, OP\_CallI), ProfiledCallI)

DEF3\_WMS(CALL\_FLAGS\_None, ProfiledReturnTypeCallIExtended, PROFILEDOP(OP\_ProfiledReturnTypeCallIExtended, OP\_CallIExtended), ProfiledCallIExtended)

DEF3\_WMS(CALL\_FLAGS\_None, ProfiledReturnTypeCallIExtendedFlags, PROFILEDOP(OP\_ProfiledReturnTypeCallIExtendedFlags, OP\_CallIExtendedFlags), ProfiledCallIExtendedFlags)

DEF3\_WMS(CALL\_FLAGS\_None, ProfiledReturnTypeCallIFlags, PROFILEDOP(OP\_ProfiledReturnTypeCallIFlags, OP\_CallIFlags), ProfiledCallIFlags)

EXDEF2\_WMS(A1toA1Mem, Conv\_Str, JavascriptConversion::ToString)

DEF2\_WMS(A1toA1Mem, Conv\_Obj, JavascriptOperators::ToObject)

EXDEF2\_WMS(A1toA1Mem, NewWithObject, JavascriptOperators::ToWithObject)

DEF2\_WMS(A1toA1Mem, Conv\_Num, JavascriptOperators::ToNumber)

DEF2\_WMS(A1toA1Mem, Incr\_A, JavascriptMath::Increment)

DEF2\_WMS(A1toA1Mem, Decr\_A, JavascriptMath::Decrement)

DEF2\_WMS(A1toA1Mem, Neg\_A, JavascriptMath::Negate)

DEF2\_WMS(A1toA1Mem, Not\_A, JavascriptMath::Not)

DEF2\_WMS(A1toA1Mem, Typeof, JavascriptOperators::Typeof)

DEF2\_WMS(A1toA1Mem, Delete\_A, JavascriptOperators::Delete)

DEF2\_WMS(GET\_ELEM\_IMem, TypeofElem, JavascriptOperators::TypeofElem)

DEF2\_WMS(A3toA1Mem, Concat3, JavascriptOperators::Concat3)

DEF2\_WMS(A2I1toA1Mem, NewConcatStrMulti, JavascriptOperators::NewConcatStrMulti)

DEF2\_WMS(A2I1toXXMem, SetConcatStrMultiItem, JavascriptOperators::SetConcatStrMultiItem)

DEF2\_WMS(A3I1toXXMem, SetConcatStrMultiItem2, JavascriptOperators::SetConcatStrMultiItem2)

DEF2\_WMS(A2toA1Mem, Add\_A, JavascriptMath::Add)

DEF2\_WMS(A2toA1Mem, Div\_A, JavascriptMath::Divide)

DEF2\_WMS(A2toA1MemProfiled, ProfiledDiv\_A, PROFILEDOP(ProfiledDivide<true>, ProfiledDivide<false>))

DEF2\_WMS(A2toA1Mem, Mul\_A, JavascriptMath::Multiply)

DEF2\_WMS(A2toA1Mem, Expo\_A, JavascriptMath::Exponentiation)

DEF2\_WMS(A2toA1Mem, Rem\_A, JavascriptMath::Modulus)

DEF2\_WMS(A2toA1MemProfiled, ProfiledRem\_A, PROFILEDOP(ProfileModulus<true>, ProfileModulus<false>))

DEF2\_WMS(A2toA1Mem, Sub\_A, JavascriptMath::Subtract)

DEF2\_WMS(A2toA1Mem, And\_A, JavascriptMath::And)

DEF2\_WMS(A2toA1Mem, Or\_A, JavascriptMath::Or)

DEF2\_WMS(A2toA1Mem, Xor\_A, JavascriptMath::Xor)

DEF2\_WMS(A2toA1Mem, Shl\_A, JavascriptMath::ShiftLeft)

DEF2\_WMS(A2toA1Mem, Shr\_A, JavascriptMath::ShiftRight)

DEF2\_WMS(A2toA1Mem, ShrU\_A, JavascriptMath::ShiftRightU)

DEF2\_WMS(CMMem, CmEq\_A, JavascriptOperators::Equal)

DEF2\_WMS(CMMem, CmGt\_A, JavascriptOperators::Greater)

DEF2\_WMS(CMMem, CmGe\_A, JavascriptOperators::GreaterEqual)

DEF2\_WMS(CMMem, CmLt\_A, JavascriptOperators::Less)

DEF2\_WMS(CMMem, CmLe\_A, JavascriptOperators::LessEqual)

DEF2\_WMS(CMMem, CmNeq\_A, JavascriptOperators::NotEqual)

DEF2\_WMS(CMMem, CmSrEq\_A, JavascriptOperators::StrictEqual)

DEF2\_WMS(CMMem, CmSrNeq\_A, JavascriptOperators::NotStrictEqual)

DEF2\_WMS(FALLTHROUGH, BeginSwitch, /\* Common case with Ld\_A \*/)

DEF2\_WMS(FALLTHROUGH, InitConst, /\* Common case with Ld\_A \*/)

DEF2\_WMS(A1toA1\_ALLOW\_STACK, Ld\_A, OP\_Ld\_A)

DEF2\_WMS(INNERtoA1, LdInnerScope, OP\_Ld\_A)

DEF2\_WMS(XXtoA1, LdLocalObj, OP\_LdLocalObj)

EXDEF2\_WMS(A1toA1\_ALLOW\_STACK, UnwrapWithObj, JavascriptOperators::OP\_UnwrapWithObj)

EXDEF2\_WMS(A2toXX, SetComputedNameVar, JavascriptOperators::OP\_SetComputedNameVar)

DEF2\_WMS(A1toXX\_ALLOW\_STACK, ChkUndecl, OP\_ChkUndecl)

DEF2\_WMS(XXtoA1, InitUndecl, OP\_InitUndecl)

DEF2\_WMS(ELEM\_RtU\_to\_XX, EnsureNoRootFld, OP\_EnsureNoRootProperty)

DEF2\_WMS(ELEM\_RtU\_to\_XX, EnsureNoRootRedeclFld, OP\_EnsureNoRootRedeclProperty)

DEF2\_WMS(ELEM\_C2\_to\_XX, ScopedEnsureNoRedeclFld, OP\_ScopedEnsureNoRedeclProperty)

DEF2\_WMS(A1toA1Profiled, ProfiledBeginSwitch, PROFILEDOP(ProfiledSwitch<true>, ProfiledSwitch<false>))

DEF2\_WMS(XXtoA1Mem, LdC\_A\_Null, JavascriptOperators::OP\_LdNull)

DEF2\_WMS(XXtoA1, ArgIn0, OP\_ArgIn0)

DEF2\_WMS(CUSTOM\_ArgNoSrc, ArgOut\_Env, OP\_ArgOut\_Env)

#if DBG

DEF2\_WMS(CUSTOM\_L\_Arg, ArgOut\_ANonVar, OP\_ArgOut\_ANonVar)

#else

DEF2\_WMS(FALLTHROUGH, ArgOut\_ANonVar, /\* Common case with ArgOUt\_A in fre build \*/)

#endif

DEF2\_WMS(CUSTOM\_L\_Arg, ArgOut\_A, OP\_ArgOut\_A)

DEF3\_WMS(CUSTOM\_L\_Arg2, ProfiledArgOut\_A, PROFILEDOP(OP\_ProfiledArgOut\_A, OP\_ArgOut\_A), ProfiledArg)

DEF3\_WMS(CUSTOM\_L\_Value, LdFld, OP\_GetProperty, ElementCP)

DEF3\_WMS(CUSTOM\_L\_Value, LdLocalFld, OP\_GetLocalProperty, ElementP)

EXDEF3\_WMS(CUSTOM\_L\_Value, LdSuperFld, OP\_GetSuperProperty, ElementC2)

DEF3\_WMS(CUSTOM\_L\_Value, LdFldForTypeOf, OP\_GetPropertyForTypeOf, ElementCP)

EXDEF3\_WMS(CUSTOM\_L\_Value, LdRootFldForTypeOf, OP\_GetRootPropertyForTypeOf, ElementRootCP)

DEF3\_WMS(CUSTOM\_L\_Value, LdFldForCallApplyTarget, OP\_GetProperty, ElementCP)

DEF3\_WMS(CUSTOM\_L\_Value, ProfiledLdFld, PROFILEDOP(OP\_ProfiledGetProperty, OP\_GetProperty), ElementCP)

DEF3\_WMS(CUSTOM\_L\_Value, ProfiledLdLocalFld, PROFILEDOP(OP\_ProfiledGetLocalProperty, OP\_GetLocalProperty), ElementP)

EXDEF3\_WMS(CUSTOM\_L\_Value, ProfiledLdSuperFld, PROFILEDOP(OP\_ProfiledGetSuperProperty, OP\_GetSuperProperty), ElementC2)

DEF3\_WMS(CUSTOM\_L\_Value, ProfiledLdFldForTypeOf, PROFILEDOP(OP\_ProfiledGetPropertyForTypeOf, OP\_GetPropertyForTypeOf), ElementCP)

EXDEF3\_WMS(CUSTOM\_L\_Value, ProfiledLdRootFldForTypeOf, PROFILEDOP(OP\_ProfiledGetRootPropertyForTypeOf, OP\_GetRootPropertyForTypeOf), ElementRootCP)

DEF3\_WMS(CUSTOM\_L\_Value, ProfiledLdFldForCallApplyTarget,PROFILEDOP(OP\_ProfiledGetPropertyCallApplyTarget, OP\_GetProperty), ElementCP)

DEF3\_WMS(CUSTOM\_L\_Value, LdRootFld, OP\_GetRootProperty, ElementRootCP)

DEF3\_WMS(CUSTOM\_L\_Value, ProfiledLdRootFld, PROFILEDOP(OP\_ProfiledGetRootProperty, OP\_GetRootProperty), ElementRootCP)

DEF3\_WMS(CUSTOM\_L\_Value, LdMethodFld, OP\_GetMethodProperty, ElementCP)

DEF3\_WMS(CUSTOM\_L\_Value, ProfiledLdMethodFld, PROFILEDOP(OP\_ProfiledGetMethodProperty, OP\_GetMethodProperty), ElementCP)

EXDEF3\_WMS(CUSTOM\_L\_Value, LdLocalMethodFld, OP\_GetLocalMethodProperty, ElementP)

EXDEF3\_WMS(CUSTOM\_L\_Value, ProfiledLdLocalMethodFld, PROFILEDOP(OP\_ProfiledGetLocalMethodProperty, OP\_GetLocalMethodProperty), ElementP)

DEF3\_WMS(CUSTOM\_L\_Value, LdRootMethodFld, OP\_GetRootMethodProperty, ElementRootCP)

DEF3\_WMS(CUSTOM\_L\_Value, ProfiledLdRootMethodFld, PROFILEDOP(OP\_ProfiledGetRootMethodProperty, OP\_GetRootMethodProperty), ElementRootCP)

DEF3\_WMS(CUSTOM\_L\_Value, DeleteFld, OP\_DeleteFld, ElementC)

EXDEF3\_WMS(CUSTOM\_L\_Value, DeleteLocalFld, OP\_DeleteLocalFld, ElementU)

DEF3\_WMS(CUSTOM\_L\_Value, DeleteRootFld, OP\_DeleteRootFld, ElementC)

DEF3\_WMS(CUSTOM\_L\_Value, DeleteFldStrict, OP\_DeleteFldStrict, ElementC)

DEF3\_WMS(CUSTOM\_L\_Value, DeleteRootFldStrict, OP\_DeleteRootFldStrict, ElementC)

DEF3\_WMS(CUSTOM, StFld, OP\_SetProperty, ElementCP)

DEF3\_WMS(CUSTOM, StLocalFld, OP\_SetLocalProperty, ElementP)

EXDEF3\_WMS(CUSTOM\_L\_Value, StSuperFld, OP\_SetSuperProperty, ElementC2)

DEF3\_WMS(CUSTOM, ProfiledStFld, PROFILEDOP(OP\_ProfiledSetProperty, OP\_SetProperty), ElementCP)

DEF3\_WMS(CUSTOM, ProfiledStLocalFld, PROFILEDOP(OP\_ProfiledSetLocalProperty, OP\_SetLocalProperty), ElementP)

EXDEF3\_WMS(CUSTOM\_L\_Value, ProfiledStSuperFld, PROFILEDOP(OP\_ProfiledSetSuperProperty, OP\_SetSuperProperty), ElementC2)

DEF3\_WMS(CUSTOM, StRootFld, OP\_SetRootProperty, ElementRootCP)

DEF3\_WMS(CUSTOM, ProfiledStRootFld, PROFILEDOP(OP\_ProfiledSetRootProperty, OP\_SetRootProperty), ElementRootCP)

DEF3\_WMS(CUSTOM, StFldStrict, OP\_SetPropertyStrict, ElementCP)

DEF3\_WMS(CUSTOM, ProfiledStFldStrict, PROFILEDOP(OP\_ProfiledSetPropertyStrict, OP\_SetPropertyStrict), ElementCP)

DEF3\_WMS(CUSTOM, StRootFldStrict, OP\_SetRootPropertyStrict, ElementRootCP)

DEF3\_WMS(CUSTOM, ProfiledStRootFldStrict, PROFILEDOP(OP\_ProfiledSetRootPropertyStrict, OP\_SetRootPropertyStrict), ElementRootCP)

DEF3\_WMS(CUSTOM, InitFld, OP\_InitProperty, ElementCP)

DEF3\_WMS(CUSTOM, ProfiledInitFld, PROFILEDOP(OP\_ProfiledInitProperty, OP\_InitProperty), ElementCP)

DEF3\_WMS(CUSTOM, InitLocalFld, OP\_InitLocalProperty, ElementP)

DEF3\_WMS(CUSTOM, ProfiledInitLocalFld, PROFILEDOP(OP\_ProfiledInitLocalProperty, OP\_InitLocalProperty), ElementP)

DEF3\_WMS(CUSTOM, InitRootFld, OP\_InitRootProperty, ElementRootCP)

DEF3\_WMS(CUSTOM, ProfiledInitRootFld, PROFILEDOP(OP\_ProfiledInitRootProperty, OP\_InitRootProperty), ElementRootCP)

DEF3\_WMS(CUSTOM, InitUndeclLetFld, OP\_InitUndeclLetProperty, ElementPIndexed)

EXDEF3\_WMS(CUSTOM, InitUndeclLocalLetFld, OP\_InitUndeclLocalLetProperty, ElementP)

DEF3\_WMS(CUSTOM, InitUndeclConstFld, OP\_InitUndeclConstProperty, ElementPIndexed)

EXDEF3\_WMS(CUSTOM, InitUndeclLocalConstFld, OP\_InitUndeclLocalConstProperty, ElementP)

DEF3\_WMS(CUSTOM, InitLetFld, OP\_InitLetFld, ElementCP)

EXDEF3\_WMS(CUSTOM, InitLocalLetFld, OP\_InitLocalLetFld, ElementP)

EXDEF3\_WMS(CUSTOM, InitInnerFld, OP\_InitInnerFld, ElementPIndexed)

EXDEF3\_WMS(CUSTOM, InitInnerLetFld, OP\_InitInnerLetFld, ElementPIndexed)

DEF3\_WMS(CUSTOM, InitRootLetFld, OP\_InitRootLetFld, ElementRootCP)

DEF3\_WMS(CUSTOM, InitConstFld, OP\_InitConstFld, ElementCP)

DEF3\_WMS(CUSTOM, InitRootConstFld, OP\_InitRootConstFld, ElementRootCP)

DEF2\_WMS(ELEM\_RtU\_to\_XX, InitUndeclRootLetFld, OP\_InitUndeclRootLetProperty)

DEF2\_WMS(ELEM\_RtU\_to\_XX, InitUndeclRootConstFld, OP\_InitUndeclRootConstProperty)

EXDEF3\_WMS(CUSTOM, InitUndeclConsoleLetFld, OP\_InitUndeclConsoleLetProperty, ElementScopedU)

EXDEF3\_WMS(CUSTOM, InitUndeclConsoleConstFld, OP\_InitUndeclConsoleConstProperty, ElementScopedU)

DEF3\_WMS(CUSTOM, InitClassMember, OP\_InitClassMember, ElementCP)

EXDEF3\_WMS(CUSTOM, InitClassMemberComputedName,OP\_InitClassMemberComputedName, ElementI)

EXDEF3\_WMS(CUSTOM, InitClassMemberSet, OP\_InitClassMemberSet, ElementC)

EXDEF3\_WMS(CUSTOM, InitClassMemberGetComputedName, OP\_InitClassMemberGetComputedName, ElementI)

EXDEF3\_WMS(CUSTOM, InitClassMemberGet, OP\_InitClassMemberGet, ElementC)

EXDEF3\_WMS(CUSTOM, InitClassMemberSetComputedName, OP\_InitClassMemberSetComputedName, ElementI)

EXDEF2\_WMS(BRB, BrOnClassConstructor, OP\_BrOnClassConstructor)

DEF3\_WMS(GET\_ELEM\_LOCALSLOTNonVar,LdLocalSlot, OP\_LdSlot, ElementSlotI1)

DEF3\_WMS(GET\_ELEM\_INNERSLOTNonVar,LdInnerSlot, OP\_LdInnerSlot, ElementSlotI2)

EXDEF3\_WMS(GET\_ELEM\_INNERSLOTNonVar,LdInnerObjSlot, OP\_LdInnerObjSlot, ElementSlotI2)

DEF3\_WMS(GET\_ELEM\_ENVSLOTNonVar, LdEnvSlot, OP\_LdEnvSlot, ElementSlotI2)

DEF3\_WMS(GET\_ELEM\_ENVSLOTNonVar, LdEnvObj, OP\_LdEnvObj, ElementSlotI1)

EXDEF3\_WMS(GET\_ELEM\_ENVSLOTNonVar, LdEnvObjSlot, OP\_LdEnvObjSlot, ElementSlotI2)

DEF3\_WMS(GET\_ELEM\_SLOTNonVar, ProfiledLdSlot, PROFILEDOP(OP\_ProfiledLdSlot, OP\_LdSlot), ProfiledElementSlot)

DEF3\_WMS(GET\_ELEM\_INNERSLOTNonVar,ProfiledLdInnerSlot, PROFILEDOP(OP\_ProfiledLdInnerSlot, OP\_LdInnerSlot), ProfiledElementSlotI2)

EXDEF3\_WMS(GET\_ELEM\_INNERSLOTNonVar,ProfiledLdInnerObjSlot, PROFILEDOP(OP\_ProfiledLdInnerObjSlot, OP\_LdInnerObjSlot), ProfiledElementSlotI2)

DEF3\_WMS(GET\_ELEM\_LOCALSLOTNonVar,ProfiledLdLocalSlot, PROFILEDOP(OP\_ProfiledLdSlot, OP\_LdSlot), ProfiledElementSlotI1)

DEF3\_WMS(GET\_ELEM\_ENVSLOTNonVar, ProfiledLdEnvSlot, PROFILEDOP(OP\_ProfiledLdEnvSlot, OP\_LdEnvSlot), ProfiledElementSlotI2)

EXDEF3\_WMS(GET\_ELEM\_ENVSLOTNonVar, ProfiledLdEnvObjSlot, PROFILEDOP(OP\_ProfiledLdEnvObjSlot, OP\_LdEnvObjSlot), ProfiledElementSlotI2)

EXDEF3\_WMS(GET\_ELEM\_SLOTNonVar, LdObjSlot, OP\_LdObjSlot, ElementSlot)

EXDEF3\_WMS(GET\_ELEM\_SLOTNonVar, ProfiledLdObjSlot, PROFILEDOP(OP\_ProfiledLdObjSlot, OP\_LdObjSlot), ProfiledElementSlot)

EXDEF3\_WMS(GET\_ELEM\_LOCALSLOTNonVar,LdLocalObjSlot, OP\_LdObjSlot, ElementSlotI1)

EXDEF3\_WMS(GET\_ELEM\_LOCALSLOTNonVar,ProfiledLdLocalObjSlot, PROFILEDOP(OP\_ProfiledLdObjSlot, OP\_LdObjSlot), ProfiledElementSlotI1)

DEF2\_WMS(SET\_ELEM\_LOCALSLOTNonVar,StLocalSlot, OP\_StSlot)

EXDEF2\_WMS(SET\_ELEM\_LOCALSLOTNonVar,StLocalSlotChkUndecl, OP\_StSlotChkUndecl)

DEF2\_WMS(SET\_ELEM\_INNERSLOTNonVar,StInnerSlot, OP\_StSlot)

EXDEF2\_WMS(SET\_ELEM\_INNERSLOTNonVar,StInnerObjSlot, OP\_StObjSlot)

DEF2\_WMS(SET\_ELEM\_ENVSLOTNonVar, StEnvSlot, OP\_StEnvSlot)

EXDEF2\_WMS(SET\_ELEM\_ENVSLOTNonVar, StEnvSlotChkUndecl, OP\_StEnvSlotChkUndecl)

EXDEF2\_WMS(SET\_ELEM\_SLOTNonVar, StObjSlot, OP\_StObjSlot)

EXDEF2\_WMS(SET\_ELEM\_LOCALSLOTNonVar,StLocalObjSlot, OP\_StObjSlot)

EXDEF2\_WMS(SET\_ELEM\_LOCALSLOTNonVar,StLocalObjSlotChkUndecl, OP\_StObjSlotChkUndecl)

EXDEF2\_WMS(SET\_ELEM\_ENVSLOTNonVar, StEnvObjSlot, OP\_StEnvObjSlot)

EXDEF2\_WMS(SET\_ELEM\_SLOTNonVar, StObjSlotChkUndecl, OP\_StObjSlotChkUndecl)

EXDEF2\_WMS(SET\_ELEM\_ENVSLOTNonVar, StEnvObjSlotChkUndecl, OP\_StEnvObjSlotChkUndecl)

DEF3\_WMS(CUSTOM\_L\_Value, LdElemI\_A, OP\_GetElementI, ElementI)

DEF3\_WMS(CUSTOM\_L\_Value, ProfiledLdElemI\_A, PROFILEDOP(OP\_ProfiledGetElementI, OP\_GetElementI), ProfiledElementI)

DEF2\_WMS(GET\_ELEM\_IMem, LdMethodElem, JavascriptOperators::OP\_GetMethodElement)

DEF3\_WMS(CUSTOM, StElemI\_A, OP\_SetElementI, ElementI)

DEF3\_WMS(CUSTOM, StElemI\_A\_Strict, OP\_SetElementIStrict, ElementI)

DEF3\_WMS(CUSTOM\_L\_Value, ProfiledStElemI\_A, PROFILEDOP(OP\_ProfiledSetElementI, OP\_SetElementI), ProfiledElementI)

DEF3\_WMS(CUSTOM\_L\_Value, ProfiledStElemI\_A\_Strict, PROFILEDOP(OP\_ProfiledSetElementIStrict, OP\_SetElementIStrict), ProfiledElementI)

DEF3\_WMS(CUSTOM, StArrItemI\_CI4, OP\_SetArrayItemI\_CI4, ElementUnsigned1)

DEF2\_WMS(FALLTHROUGH, StArrInlineItem\_CI4, /\*Common case with StArrItemC\_CI4 \*/)

DEF3\_WMS(CUSTOM, StArrItemC\_CI4, OP\_SetArrayItemC\_CI4, ElementUnsigned1)

DEF3\_WMS(CUSTOM\_L\_R0, LdArrHead, OP\_LdArrayHeadSegment, Reg2)

DEF3\_WMS(CUSTOM, StArrSegItem\_CI4, OP\_SetArraySegmentItem\_CI4, ElementUnsigned1)

DEF3 (CUSTOM, StArrSegItem\_A, OP\_SetArraySegmentVars, Auxiliary)

DEF3\_WMS(CALL, NewScObject, OP\_NewScObject, CallI)

DEF3\_WMS(CUSTOM\_L\_R0, NewScObjectNoCtorFull, OP\_NewScObjectNoCtorFull, Reg2)

EXDEF2\_WMS(A1toA1Mem, LdCustomSpreadIteratorList, JavascriptOperators::OP\_LdCustomSpreadIteratorList)

EXDEF3\_WMS(CALL, NewScObjectSpread, OP\_NewScObjectSpread, CallIExtended)

DEF3\_WMS(CALL, NewScObjArray, OP\_NewScObjArray, CallI)

DEF3\_WMS(CALL, NewScObjArraySpread, OP\_NewScObjArraySpread, CallIExtended)

DEF3\_WMS(CALL, ProfiledNewScObject, PROFILEDOP(OP\_ProfiledNewScObject, OP\_NewScObject), ProfiledCallI)

EXDEF3\_WMS(CALL, ProfiledNewScObjectSpread, PROFILEDOP(OP\_ProfiledNewScObjectSpread, OP\_NewScObjectSpread), ProfiledCallIExtended)

DEF3\_WMS(CALL, ProfiledNewScObjectWithICIndex, PROFILEDOP(OP\_ProfiledNewScObjectWithICIndex, OP\_NewScObject), ProfiledCallIWithICIndex)

DEF3\_WMS(CALL, ProfiledNewScObjArray, PROFILEDOP(OP\_ProfiledNewScObjArray, OP\_ProfiledNewScObjArray\_NoProfile), Profiled2CallI)

DEF3\_WMS(CALL, ProfiledNewScObjArraySpread,PROFILEDOP(OP\_ProfiledNewScObjArraySpread, OP\_ProfiledNewScObjArraySpread\_NoProfile), Profiled2CallIExtended)

DEF2\_WMS(XXtoA1NonVar, LdArgCnt, OP\_LdArgCnt)

DEF3\_WMS(CUSTOM\_L\_R0, LdLen\_A, OP\_LdLen, Reg2)

DEF3\_WMS(CUSTOM\_L\_R0, ProfiledLdLen\_A, PROFILEDOP(OP\_ProfiledLdLen, OP\_LdLen), ProfiledReg2)

DEF2\_WMS(XXtoA1Mem, LdUndef, JavascriptOperators::OP\_LdUndef)

DEF2\_WMS(XXtoA1Mem, LdNaN, JavascriptOperators::OP\_LdNaN)

DEF2\_WMS(XXtoA1Mem, LdInfinity, JavascriptOperators::OP\_LdInfinity)

DEF2\_WMS(XXtoA1Mem, LdTrue, JavascriptBoolean::OP\_LdTrue)

DEF2\_WMS(XXtoA1Mem, LdFalse, JavascriptBoolean::OP\_LdFalse)

DEF2\_WMS(A1I1toA1Mem, LdThis, JavascriptOperators::OP\_GetThisNoFastPath)

EXDEF2\_WMS(XXtoA1Mem, LdSuper, OP\_LdSuper)

EXDEF2\_WMS(XXtoA1Mem, LdSuperCtor, OP\_LdSuperCtor)

EXDEF2\_WMS(XXtoA1Mem, ScopedLdSuper, OP\_ScopedLdSuper)

EXDEF2\_WMS(XXtoA1Mem, ScopedLdSuperCtor, OP\_ScopedLdSuperCtor)

EXDEF2\_WMS(A2toXX, SetHomeObj, JavascriptOperators::OP\_SetHomeObj)

DEF2\_WMS(A1toA1Mem, StrictLdThis, JavascriptOperators::OP\_StrictGetThis)

DEF2\_WMS(A1I1toA1Mem, ProfiledLdThis, PROFILEDOP(OP\_ProfiledLdThis, JavascriptOperators::OP\_GetThisNoFastPath))

DEF2\_WMS(A1toA1Mem, ProfiledStrictLdThis, PROFILEDOP(OP\_ProfiledStrictLdThis, JavascriptOperators::OP\_StrictGetThis))

DEF2\_WMS(XXtoA1Mem, LdHeapArgsCached, OP\_LdHeapArgsCached)

EXDEF2\_WMS(XXtoA1Mem, LdLetHeapArgsCached, OP\_LdLetHeapArgsCached)

EXDEF2\_WMS(XXtoA1NonVar, LdStackArgPtr, OP\_LdStackArgPtr)

EXDEF3\_WMS(CUSTOM, InitSetFld, OP\_InitSetFld, ElementC)

EXDEF3\_WMS(CUSTOM, InitGetFld, OP\_InitGetFld, ElementC)

EXDEF3\_WMS(CUSTOM, InitSetElemI, OP\_InitSetElemI, ElementI)

EXDEF3\_WMS(CUSTOM, InitGetElemI, OP\_InitGetElemI, ElementI)

EXDEF3\_WMS(CUSTOM, InitComputedProperty, OP\_InitComputedProperty, ElementI)

EXDEF3\_WMS(CUSTOM, InitProto, OP\_InitProto, ElementC)

DEF3\_WMS(CUSTOM, LdElemUndefScoped, OP\_LdElementUndefinedScoped, ElementScopedU)

DEF3\_WMS(CUSTOM\_L\_R0, LdFuncExpr, OP\_LdFunctionExpression, Reg1)

DEF3\_WMS(CUSTOM, StFuncExpr, OP\_StFunctionExpression, ElementC)

DEF3\_WMS(CUSTOM, StLocalFuncExpr, OP\_StLocalFunctionExpression, ElementU)

EXDEF3\_WMS(CUSTOM\_L\_R0, LdNewTarget, OP\_LdNewTarget, Reg1)

EXDEF2 (EMPTY, ChkNewCallFlag, OP\_ChkNewCallFlag)

DEF2\_WMS(U1toINNERMemNonVar, NewBlockScope, JavascriptOperators::OP\_NewBlockScope)

DEF3\_WMS(CUSTOM, CloneBlockScope, OP\_CloneBlockScope, Unsigned1)

DEF2\_WMS(U1toINNERMemNonVar, NewPseudoScope, JavascriptOperators::OP\_NewPseudoScope)

DEF3\_WMS(CUSTOM\_L\_Value, NewStackScFunc, OP\_NewStackScFunc, ElementSlotI1)

DEF2\_WMS(GET\_SLOT\_FB, NewScFunc, ScriptFunction::OP\_NewScFunc)

DEF2\_WMS(GET\_SLOT\_FB, NewScGenFunc, JavascriptGeneratorFunction::OP\_NewScGenFunc)

EXDEF3\_WMS(CUSTOM\_L\_Value, NewInnerStackScFunc, OP\_NewInnerStackScFunc, ElementSlot)

EXDEF2\_WMS(GET\_ELEM\_SLOT\_FB, NewInnerScFunc, ScriptFunction::OP\_NewScFunc)

EXDEF2\_WMS(GET\_ELEM\_SLOT\_FB, NewInnerScGenFunc, JavascriptGeneratorFunction::OP\_NewScGenFunc)

DEF2\_WMS(A1toA1MemNonVar, GetForInEnumerator, JavascriptOperators::OP\_GetForInEnumerator)

DEF3\_WMS(A1toXXMemNonVar, ReleaseForInEnumerator, JavascriptOperators::OP\_ReleaseForInEnumerator, ForInObjectEnumerator \*)

DEF2\_WMS(A1toXXMem, Throw, JavascriptExceptionOperators::OP\_Throw)

DEF2\_WMS(XXtoA1NonVar, LdArgumentsFromFrame, OP\_LdArgumentsFromFrame)

DEF2\_WMS(A1toA1MemNonVar, LdHeapArguments, OP\_LdHeapArguments)

DEF2\_WMS(A1toA1MemNonVar, LdLetHeapArguments, OP\_LdLetHeapArguments)

DEF2\_WMS(A2toA1MemNonVar, LdInnerFrameDisplay, OP\_LdInnerFrameDisplay)

DEF2\_WMS(A1toA1MemNonVar, LdInnerFrameDisplayNoParent,OP\_LdInnerFrameDisplayNoParent)

DEF2\_WMS(A1INNERtoA1MemNonVar, LdIndexedFrameDisplay, OP\_LdFrameDisplay)

DEF2\_WMS(XXINNERtoA1MemNonVar, LdIndexedFrameDisplayNoParent,OP\_LdFrameDisplayNoParent<true>)

DEF2\_WMS(A2toXXMemNonVar, LdFuncExprFrameDisplay, OP\_LdFuncExprFrameDisplaySetLocal)

DEF3\_WMS(CUSTOM\_L\_R0, IsInst, OP\_IsInst, Reg3C)

DEF2\_WMS(A2toA1Mem, IsIn, JavascriptOperators::IsIn)

DEF3\_WMS(CUSTOM\_L\_Value, ScopedLdFld, OP\_GetPropertyScoped, ElementP)

EXDEF3\_WMS(CUSTOM\_L\_Value, ScopedLdFldForTypeOf, OP\_GetPropertyForTypeOfScoped, ElementP)

DEF3\_WMS(CUSTOM\_L\_Value, ScopedLdMethodFld, OP\_GetMethodPropertyScoped, ElementCP)

DEF3\_WMS(CUSTOM, ScopedStFld, OP\_SetPropertyScoped, ElementP)

EXDEF3\_WMS(CUSTOM, ConsoleScopedStFld, OP\_ConsoleSetPropertyScoped, ElementP)

DEF3\_WMS(CUSTOM, ScopedStFldStrict, OP\_SetPropertyScopedStrict, ElementP)

DEF2\_WMS(GET\_ELEM\_IMem, DeleteElemI\_A, JavascriptOperators::OP\_DeleteElementI)

DEF2\_WMS(GET\_ELEM\_IMem\_Strict, DeleteElemIStrict\_A, JavascriptOperators::OP\_DeleteElementI)

DEF3\_WMS(CUSTOM\_L\_Value, ScopedLdInst, OP\_ScopedLdInst, ElementScopedC2)

DEF3\_WMS(CUSTOM, ScopedInitFunc, OP\_ScopedInitFunc, ElementScopedC)

DEF3\_WMS(CUSTOM\_L\_Value, ScopedDeleteFld, OP\_ScopedDeleteFld, ElementScopedC)

DEF3\_WMS(CUSTOM\_L\_Value, ScopedDeleteFldStrict, OP\_ScopedDeleteFldStrict, ElementScopedC)

DEF3\_WMS(CUSTOM, LdElemUndef, OP\_LdElementUndefined, ElementU)

EXDEF3\_WMS(CUSTOM, LdLocalElemUndef, OP\_LdLocalElementUndefined, ElementRootU)

DEF2\_WMS(XXtoA1, NewScObjectSimple, OP\_NewScObjectSimple)

DEF3 (CUSTOM, NewScObject\_A, OP\_NewScObject\_A, Auxiliary)

DEF3 (CUSTOM, NewScObjectLiteral, OP\_NewScObjectLiteral, Auxiliary)

DEF3 (CUSTOM\_L\_R0, LdPropIds, OP\_LdPropIds, Auxiliary)

DEF3 (CUSTOM, InitCachedFuncs, OP\_InitCachedFuncs, AuxNoReg)

DEF2\_WMS(LOCALI1toA1, GetCachedFunc, OP\_GetCachedFunc)

DEF2\_WMS(EnvU1toXX, InvalCachedScope, JavascriptOperators::OP\_InvalidateCachedScope)

DEF3 (CUSTOM, CommitScope, OP\_CommitScope, AuxNoReg)

DEF2\_WMS(A1I2toXXNonVar\_FuncBody, NewInnerScopeSlots, OP\_NewInnerScopeSlots)

DEF3\_WMS(CUSTOM, CloneInnerScopeSlots, OP\_CloneInnerScopeSlots, Unsigned1)

DEF3\_WMS(CUSTOM\_L\_R0, NewScArray, OP\_NewScArray, Reg1Unsigned1)

DEF2\_WMS(U1toA1, NewScArrayWithMissingValues,JavascriptArray::OP\_NewScArrayWithMissingValues)

DEF3 (CUSTOM\_L\_R0, NewScIntArray, OP\_NewScIntArray, Auxiliary)

DEF3 (CUSTOM\_L\_R0, NewScFltArray, OP\_NewScFltArray, Auxiliary)

DEF3\_WMS(CUSTOM\_L\_R0, ProfiledNewScArray, PROFILEDOP(OP\_ProfiledNewScArray, OP\_ProfiledNewScArray\_NoProfile), ProfiledReg1Unsigned1)

DEF3 (CUSTOM\_L\_R0, ProfiledNewScIntArray, PROFILEDOP(OP\_ProfiledNewScIntArray, OP\_NewScIntArray), ProfiledAuxiliary)

DEF3 (CUSTOM\_L\_R0, ProfiledNewScFltArray, PROFILEDOP(OP\_ProfiledNewScFltArray, OP\_NewScFltArray), ProfiledAuxiliary)

DEF2\_WMS(RegextoA1, NewRegEx, JavascriptRegExp::OP\_NewRegEx)

EXDEF3\_WMS(CUSTOM, InitClass, OP\_InitClass, Class)

DEF3\_WMS(BRBReturnP1toA1, BrOnEmpty, JavascriptOperators::OP\_BrOnEmpty, ForInObjectEnumerator \*)

DEF2 (TRY, TryCatch, OP\_TryCatch)

DEF2 (TRY, TryFinally, OP\_TryFinally)

EXDEF2\_WMS(TRYBR2, TryFinallyWithYield, OP\_TryFinallyWithYield)

EXDEF2 (EMPTY, ResumeCatch, OP\_ResumeCatch)

EXDEF2\_WMS(TRYBR2, ResumeFinally, OP\_ResumeFinally)

DEF2\_WMS(A1NonVarToA1, ResumeYield, OP\_ResumeYield)

DEF2\_WMS(A2NonVarToA1Reg, ResumeYieldStar, OP\_ResumeYield)

EXDEF2\_WMS(A2toA1Mem, AsyncSpawn, JavascriptOperators::OP\_AsyncSpawn)

EXDEF2 (W1, RuntimeTypeError, JavascriptExceptionOperators::OP\_RuntimeTypeError)

EXDEF2 (W1, RuntimeReferenceError, JavascriptExceptionOperators::OP\_RuntimeReferenceError)

DEF3 (CUSTOM\_L\_R0, SpreadArrayLiteral, OP\_SpreadArrayLiteral, Reg2Aux)

EXDEF2\_WMS(A1toXX, ObjectFreeze, JavascriptOperators::OP\_Freeze)

EXDEF3\_WMS(CUSTOM, ClearAttributes, OP\_ClearAttributes, ElementU)

DEF3\_WMS(CUSTOM, ApplyArgs, OP\_ApplyArgs, Reg5)

EXDEF3\_WMS(CUSTOM, EmitTmpRegCount, OP\_EmitTmpRegCount, Unsigned1)

#endif

// help the caller to undefine all the macros

#undef DEF2

#undef DEF3

#undef DEF2\_WMS

#undef DEF3\_WMS

#undef DEF4\_WMS

#undef EXDEF2

#undef EXDEF3

#undef EXDEF2\_WMS

#undef EXDEF3\_WMS

#undef EXDEF4\_WMS