CINCOM Technical Note

VisualWorks Bytecode Set

Software Release 7.6
Last Updated: November 12, 2007

This document describes the VisualWorks bytecode set introduced in version 5i, and the expected behavior of the bytecode set, updated for the noted software release. This technical note does not go into any detail regarding motivation or design decisions.

The following bytecodes are described as Short (using 1 byte), or two-, three-, or four-byte bytecodes. For the purposes of this document, the bytes in a bytecode will be labeled BC (i.e. the leading byte, which usually identifies which operation is wanted), B1, B2, and B3. B1 through B3 identify additional arguments which the bytecode may need to function. These four labels will be used in the comments below to show how the arguments of the bytecode are computed.

When a literal, local, or instance variable is identified by an index below, that index is always assumed to be 0-based.

When a jump instruction describes jumping to another location in the method, that location is always measured from the first instruction *following* the jump instruction.

For message send bytecodes, arguments are pushed left-to-right so that the rightmost argument is always at the top of stack.

Bytecode Groups:

Short Load Codes

Code Name	Code Value	Comment
OpLoadInst	16r00 — 16r0F (16)	Push the instance variable BC of the receiver onto the stack.
OpLoadTemp	16r10 — 16r1B (12)	Push the local at BC-16r10 onto the stack.
OpLoadLiteral	16r1C — 16r2B (16)	Push the literal found at BC-16r1C onto the stack.
—	16r2C-16r33 (8)	**Not used at this time.**
OpLoadStatic	16r34 — 16r3F (12)	Push the literal found at BC-16r34 onto the stack, and send #value.
OpNoOp	16r40	Do nothing.
—	16r41	**Not used at this time.**
—	16r42	**Not used at this time.** This bytecode has been reserved as a possible breakpoint bytecode.
OpPrimReturn	16r43	Instruction indicating that if the primitive succeeds, the method should return without running its failure code. Can only appear after OpPrimitive, and is not optional in that position.
OpLoadReceiver	16r44	Push the receiver onto the stack
OpPopLoadReceiver	16r45	Pop the top value off the stack, and push the receiver.
OpLoadNil	16r46	Push nil onto the stack
OpLoadTrue	16r47	Push true onto the stack.
OpLoadFalse	16r48	Push false onto the stack
OpLoadZero	16r49 — 16r4B (3)	Push an integer 0, 1 or 2 onto the stack, depending on the value of BC-16r49
OpStorePopTemp	16r4C — 16r53 (8)	Pop the top value off the stack and store it in the local at BC-16r4C
OpLoadThisContext	16r54	Push the current context onto the stack

Code Name	Code Value	Comment
—	16r55	**Not used at this time.**
OpPopLoadTemp	16r56 — 16r57 (2)	Pop the top value off the stack, and push onto the stack the value of the local at BC-16r56.
OpStorePopInst	16r58 — 16r5F (8)	Pop the top value off the stack and store it in the instance variable at BC-16r58 of the receiver.

Short Misc. Codes

Code Name	Code Value	Comment
OpReturnReceiver	16r60	Return from the method block or method using the receiver as the return value.
—	16r61	**Not used at this time.**
OpReturnNil	16r62	Return from the method block or method using nil as the return value.
OpReturnTrue	16r63	Return from the method block or method using true as the return value.
OpReturnFalse	16r64	Return from the method block or method using false as the return value.
OpReturn	16r65	Return from the method block or method using the top value of the stack as the return value.
OpPop	16r66	Pop the top value off the stack.
OpLoopHead	16r67	Target of a backwards (looping) branch. It is an error to have a backward branch whose target is not this instruction.

Code Name	Code Value	Comment
OpDupFirst	16r68	Duplicate the top value on the stack, in preparation to send the first message of a cascade. The duplicated value is the receiver of the cascaded messaages.
OpDupNext	16r69	Pop the top value off the stack, then duplicate the new top value of the stack. This is used in preparing to send a message that is part of a cascade, when the message is neither the first nor the last message in the cascade. In such a case, the value which is to be popped is the return value of the preceding message of the cascade, and the value being duplicated is the receiver of the cascade.
OpNoDup	16r6A	Pop the top value off the stack. The popped value is the return value of the previous message send, removed in preparation to send the last message of a cascade.

Short Control Codes

Code Name	Code Value	Comment
OpShortJump	16r6B — 16r6F (5)	Jump forward BC-16r6B+1 bytes in the method.
OpSends	16r70 — 16r7F (16)	Send a message with 0 arguments. The selector is found at literal index BC-16r70.
	16r80 — 16r8F (16)	Send a message with 1 arguments. The selector is found at literal index BC-16r80.
	16r90 — 16r97 (8)	Send a message with 2 arguments. The selector is found at literal index BC-16r90.
OpSendSelf0	16r98 — 16r9F (8)	Send a 0-argument to the receiver. The selector is the literal at BC-16r98.
OpSpecialSend	16rA0 — 16rBF (32)	Send a message, where the receiver and arguments are all on the stack. The selector and how many arguments it expects can be found in the SpecialSelectors array at logical index BC-16rA0.

Code Name	Code Value	Comment
OpShortBranchFalse	16rC0 — 16rC7 (8)	The top of the stack must be a boolean. Pop it off, and if false, jump forward BC+1-16rC0 bytes in the method.
OpSendAdd1	16rC8	Push 1 on the stack and send #+.
—	16rC9	**Not used at this time.**
—	16rCA	**Not used at this time.**

Two-Byte Misc. Codes

Code Name	Code Value	Comment
OpCopyValues	16rCB	The receiver must be a BlockClosure. Copy its B1 copiedValues onto the stack.
OpXNoCheckSend	16rCC	Send a message, where the receiver and arguments are all on the stack. The message takes B1//32 arguments, and is found at literal index B1\\32. The type of the message's receiver was known at compile time, so that the translator doesn't need to include runtime type checking.
OpXNonImmediateSend	16rCD	Send a message, where the receiver and arguments are all on the stack. The selector is the literal at B1\\32, and it takes B1//32 arguments. We don't know the type of the receiver except that it cannot be a SmallInteger or a Character.
OpXNonImmediateSpecialSend	16rCE	Send a message, where the receiver and arguments are all on the stack. The selector and how many arguments it expects can be found in the SpecialSelectors array at logical index B1. We don't know the type of the receiver except that it cannot be a SmallInteger or a Character.
OpFullBlock	16rCF	Push onto the stack a BlockClosure whose outerContext is the current context and whose method is the literal at B1.
OpXLoadInst	16rD0	Push the instance variable B1 of the receiver onto the stack.

Code Name	Code Value	Comment
OpXLoadTemp	16rD1	Push the local variable found at B1 onto the stack.
OpXLoadLiteral	16rD2	Push the literal at B1 onto the stack.
OpXLoadStatic	16rD3	Push the literal found at B1 onto the stack, and send #value.
OpCreateArray	16rD4	Create a new Array of size B1+1 and push it onto the stack.
OpConsArray	16rD5	Replace the top B1+1 values on the stack with a new Array containing those values. Their order within the Array is the same as the order in which they were pushed on the stack.
—	16rD6	**Not used at this time.**
OpLoadCharacter	16rD7	Push the Character whose Unicode value is B1 onto the stack.
OpLoadByte	16rD8	Push an integer on the stack, whose value is B1.
OpLoadLocalIndirect	16rD9	Push onto the stack the value of the instance variable B1\\16 of the local at B1//16. This allows us to load the receiver's instance variables from within a block, as well as the instance variables of Arrays created by OpCreateArray. The local is not checked to make sure it has at least B1\\16+1 slots, so this must be used carefully.
OpXStorePopInst	16rDA	Pop the top value off the stack and store it into the instance variable at B1.
OpXStorePopTemp	16rDB	Pop the top value off the stack and store it into the local at B1.
OpStorePopLocalIndirect	16rDC	Pop the top value off the stack and store it in the instance variable B1\\16 of the local at B1//16. This allows us to store into the receiver's instance variables from within a block, as well as the instance variables of Arrays created by OpCreateArray. The local is not checked to make sure it has at least B1\\16+1 slots, so this must be used carefully.
—	16rDD	**Not used at this time.**

Two-Byte Control Codes

Code Name	Code Value	Comment
OpHomeReturn	16rDE	Return the top value of the stack from the method. If the current context is a BlockContext, don't return from this context, but from the correct enclosing MethodContext, evaluating all intervening unwind protection blocks as we do so. The number of levels of indirection between the BlockContext and the MethodContext from which we are to return is B1. We find the MethodContext by tracing the chain from the current BlockContext, to that context's receiver (a BlockClosure), to the closure's outerContext, which is the next outer BlockContext (or ultimately MethodContext) in the chain.
OpXNoCheckSpecialSend	16rDF	Send a message, where the receiver and arguments are all on the stack. The selector and how many arguments it expects can be found in the SpecialSelectors array at logical index B1. The type of the message's receiver was known at compile time, so that the translator doesn't need to include runtime type checking.
OpLongJump	16rE0 — 16rE7 (8)	Jump forward (or backward if the value is negative) (BC-16rE4) * 256 + B1 bytes in the method.
OpLongBranchFalse	16rE8 — 16rEB (4)	The top of the stack must be a boolean. Pop it off, and if false, jump forward (BC-16rE8) * 256 + B1 bytes in the method.
OpLongBranchTrue	16rEC — 16rEF (4)	The top of the stack must be a boolean. Pop it off, and if true, jump forward (BC-16rE8) * 256 + B1 bytes in the method.

Code Name	Code Value	Comment
OpXSpecialSend	16rF0	Send a message, where the receiver and arguments are all on the stack. The selector and how many arguments it expects can be found in the SpecialSelectors array at logical index B1.
OpXSend	16rF1	Send a message, where the receiver and arguments are all on the stack. The selector is the literal at B1\\32, and it takes B1//32 arguments.
OpXSuper	16rF2	Send a super message to the receiver. The receiver, the mclass of the method, and the arguments are already on the stack. The selector is the literal at B1\\32, and takes B1//32 arguments.

Three-Byte Codes

Code Name	Code Value	Comment
OpXXNonImmediateSend	16rF3	Send a message, where the receiver and arguments are all on the stack. The selector is the literal at B2, and it takes B1 arguments. We don't know the type of the receiver except that it cannot be a SmallInteger or a Character.
OpXXNoCheckSend	16rF4	Send a message, where the receiver and arguments are all on the stack. The message takes B1 arguments, and is found at literal index B2. The type of the message's receiver was known at compile time, so that the translator doesn't need to include runtime type checking.
OpXLoadLocalIndirect	16rF5	Push onto the stack the value of the instance variable B2 of the local at B1. This allows us to load the receiver's instance variables from within a block, as well as the instance variables of Arrays created by OpCreateArray. The local is not checked to make sure it has at least B2+1 slots, so this must be used carefully.

Code Name	Code Value	Comment
OpXStorePopLocalIndirect	16rF6	Pop the top value off the stack and store it in the instance variable B2 of the local at B1. This allows us to store into the receiver's instance variables from within a block, as well as the instance variables of Arrays created by OpCreateArray. The local is not checked to make sure it has at least B2+1 slots, so this must be used carefully.
OpFullCopyingBlock	16rF7	Create a BlockClosure whose outerContext is the current context, whose copiedValues is an Array of the B2 top values on the stack, and whose method is the literal at B1. Optimize the case where B2 = 1, by not wrapping the copied value in an Array.
—	16rF8	**Not used at this time.**
OpLoadTwoBytes	16rF9	Push a signed integer onto the stack. If B1 < 128, the value of the integer is B1*256+B2; otherwise it's (256-B1)*256+B2. The result is an integer between -32768 and 32767.
OpXCopyingBlock	16rFA	Create a BlockClosure whose copiedValues is an Array of the B2 top values on the stack, and whose method is the literal at B1. Optimize the case where B2 = 1, by not wrapping the copied value in an Array.
—	16rFB	**Not used at this time.**
OpXXSend	16rFC	Send a message to some object. The intended receiver and the arguments are already on the stack. The selector is the literal at B2, and takes B1 arguments.
OpXXSuper	16rFD	Send a super message to the receiver. The receiver, the mclass of the method, and the arguments are already on the stack. The selector is the literal at B2, and takes B1 arguments.
OpPrimitive	16rFE	The first instruction of a method that is defined primitively. The primitive number is B1*256+B2. This must always be followed by OpPrimReturn.

Four-Byte Codes

Code Name	Code Value	Comment
—	16rFF	**Not used at this time.** The virtual image still treats this as a 3-byte code to accomodate old methods loaded from previous versions of VisualWorks.

Revision History

4 August 2003, Updated to include OpCreateArray, 16rD5, introduced in release 7.2.

13 November 2007, updated row 4, unused values, of Short Load Codes table.