Phase 4. Coder Analysis

Lawrence McKenny - 10034244 - 11lrm1
Zhiyuan Tong - 20043698 - 16zt9
Tara Carette - 20021418 - 15tkc1
Jolene Lammers - 20029503 - 16jjl4

CISC/CMPE458
James Cordy
April 19, 2020

1. Interface with Coder Phase

Location of Change

In coder.ssl under compound t-codes.

What was Changed

Add new t-code tokens:

```
tMultiply
tConcatenate
tRepeatString
tSubstring
tLength
tStringEQ
tInitialValue
tInitEnd
tCaseElse
tCaseElseEnd
```

Delete the compound T-codes:

```
trWriteString = 7
trWriteString = 7
trReadString = 13;
```

Add the new compound T-codes:

```
twofiftysix = 256
trAssignString = 101
trChrString = 102
trConcatenate = 103
trSubstring = 104
trLength = 105
trStringEqual = 107
trReadString = 108
trWriteString = 109
string = 3
trReadString = 108
trWriteString = 109
```

Reason

To add all the new Like T-codes to ensure that the list of T-codes is identical to the output tokens in the semantic.ssl. Update and add new trap codes for the new Like run time library string operation traps to the trap codes to ensure that the value are exactly the ones listed in the Phase 4 handout

2. Mixed Declarations and Statements

Location of Change

In coder.ssl under compound t-codes.

What was Changed

Move the code from “Statements” rule into the “Block” rule

And then call the Block rule in Statements

Reason
Modify the block rule to accept all the statement T-code directly in its main declaration-handling loop by coping all the alternatives in the Statements rule directly into it.

3. Initial Values

Location of Change
In coder.ssl under Block

What was Changed
Added following code:

```
        | tInitialValue:
             @OperandPushExpression
             [
                 | tInitEnd:
                     tLiteralAddress
                     oOperandPushVariable
                     [
                         | tStoreInteger:
                             oOperandSetLength(word)
                             oOperandSwap
                             @OperandAssignIntegerPopPop
                         | tStoreBoolean:
                             oOperandSetLength(byte)
                             oOperandSwap
                             @OperandAssignBooleanPopPop
                         | tStoreChar:
                             oOperandSetLength(byte)
                             oOperandSwap
                             @OperandAssignCharPopPop
                     ]
             ]
```

Reason
To handle initial value assignment

4. Choose Statement Else Clauses

Location of Change
The rule “EmitDefaultCaseAbout” of file code.ssl under compound t-codes.

What was Changed
Add new code:

```
 [
         | tCaseElse:
             % handle case else
             @Statements
             tCaseElseEnd
```

```
            oEmitCaseMergeBranch
        |*:
            % Call case abort trap (never returns)
            oOperandPushMode(mLineNum)          % ... n
            oOperandSetLength(word)
            @OperandForceToStack                %       pushl   n
            oOperandPop                         % ...
            oOperandPushMode(mTrap)
            oOperandSetValue(trCaseAbort)
            oEmitSingle(iCall)                  %       call  caseAbort
    ]
```

Delete original code:

```
% Call case abort trap (never returns)
            oOperandPushMode(mLineNum)          % ... n
            oOperandSetLength(word)
            @OperandForceToStack                %       pushl   n
            oOperandPop                         % ...
            oOperandPushMode(mTrap)
            oOperandSetValue(trCaseAbort)
            oEmitSingle(iCall)                  %       call  caseAbort
```

Reason
Add the tCaseElse case to handle the else clause.

5. String Constants, Variables and Arrays

5.1. Change relevant sizes to string

Location of Change
In coder.ssl under the rules Routine, OperandPushVariable, and OperandSubscriptCharPop

What was Changed
Change the size from byte to string

```
oOperandSetLength(string)
```

Reason
Some of the char operators from PT Pascal are adapted to work with strings. So the size bytes no longer applies and needs to be changed to strings

5.2. Delete out of date choice paths

Location of Change
In coder.ssl under the rules OperandPushExpression, OperandPushExpressionAssignPopPop

What was Changed
Deleted:

```
| tLiteralChar:
      oOperandPushChar
```

Reason

The handling of char and strings have changed and so those paths are no longer necessary for the program to run

5.3. Add subscript scaling

Location of Change

In coder.ssl in OperandCheckedSubscriptNonManifestCharPop and OperandUncheckedSubscriptNonManifestCharPop

What was Changed

Added subscripting scaling by the string size

In OperandCheckedSubscriptNonManifestCharPop:

```
% add subscripting scaling by string size
    oOperandPushMode(mManifest)
    oOperandSetLength(word)
    oOperandSetValue(eight)
    oEmitDouble(iShl)
    oOperandPop

    % Add normalized subscript to array address
    oOperandSwap                              % ... %T, arraydesc
    [ oOperandChooseMode
        | mStatic:
            % Optimize by folding array offset into array descriptor address
            oOperandPushMode(mManifest)
            oOperandSetLength(string) % switch to string size
            oOperandSetValue(twofiftysix) % switch to 256
            oOperandAddManifestValues         % ... %T, arraydesc+256, 256
            oOperandPop                       % ... %T, arraydesc+256
            @OperandForceAddressIntoTemp      %       mov     $arraydesc+256, %T2
                                              % ... %T, %T2
            % Add array address to normalized and scaled subscript
            oEmitDouble(iAdd)                 %       addl    %T2, %T
            @OperandPopAndFreeTemp            % ... %T
        | *:
            % Can't optimize
            @OperandForceAddressIntoTemp      % ... %T, %T2
            oOperandPushMode(mManifest)
            oOperandSetLength(string) % switch to size string
            oOperandSetValue(twofiftysix) % switch to 256        ... %T, %T2, 8
```

In OperandUncheckedSubscriptNonManifestCharPop:

```
 % add scaling of subscript by size string
            @OperandForceIntoTemp             %       movl    subscript, %T
            oOperandPushMode(mManifest)
            oOperandSetLength(word)
            oOperandSetValue(eight)
            oEmitDouble(iShl)
            oOperandPop
            oOperandSwap                      % ... %T, arraydesc

            % Fold lower bound into array address to avoid normalizing
            % subscript at run time
            oOperandPushArrayLowerBound       % ... %T, arraydesc, lower
```

```
            oOperandSwap                        % ... %T, lower, arraydesc
            oOperandSetMode(mManifest)          % (eliminate indirection)
            oOperandPushMode(mManifest)         % ... %T, lower, arraydesc, 8
            oOperandSetLength(string)
            oOperandSetValue(twofiftysix)
```

Reason

This operation is switched to using strings, which is larger than the previously used char. To make sure the arrays still know where the different elements are, it needs to be scaled by the size of the element

5.4. Add a tSkipString path

Location of Change

In coder.ssl in OperandPushExpressionAssignPopPop

What was Changed

Added a path so could accept the tSkipString token

```
| tSkipString:
      OperandPushExpression
      oEmitNone(iData)
      tStringData
      oEmitString
      oEmitNone(iText)
```

Reason

Strings were added to Like and so the tSkipString token is now a valid beginning to an expression

5.5. Add ability to assign strings

Location of Change

In coder.ssl in OperandAssignCharPopPop

What was Changed

Changed the entire rule to call the string trap to assign the string value

```
% change to emitting the trAssignString trap over a single byte
assignment

    % save the temp registers in case in use
    @SaveTempRegsToStack

    % send the arguments
    @OperandForceToStack
    oOperandPop

    @OperandForceAddressIntoTemp % do this before we can put it into a
stack
    @OperandForceToStack
    @OperandPopAndFreeTemp
```

```
% call assignstring trap routine
oOperandPushMode(mTrap)
oOperandSetValue(trAssignString)
oOperandSetLength(word)
oEmitSingle(iCall)
oOperandPop

% pop argument
oOperandPushMode(mStackReg)
oOperandSetLength(word)
oOperandPushMode(mManifest)
oOperandSetLength(word)
oOperandSetValue(eight)
oEmitDouble(iAdd)
oOperandPop
oOperandPop

% restore temp regs
@RestoreTempRegsFromStack;
```

Reason
Strings are more complicated than the char originally assigned through this rule and need to be handled differently in order to assign successfully (they are referred to by address and are much larger than char)

5.6. Add size string as a choice
Location of Change
In coder.ssl in OperandForceIntoTemp and OperandForceIntoStack

What was Changed
Add following new string T-code:

```
tConcatenate
tRepeatString
tSubstring
tLength
tStringEQ
```

Reason
To have t-codes that can be emitted for the new string operations.

5.7. Add definition of stringSize
Location of Change
In coder.pt under Machine word size

What was Changed
Added:

```
stringSize = 256;
```

Reason
The size of the string needs to be defined so that the semantic mechanisms can operate over strings

5.8. Add scaling for subscripting strings
Location of Change
In coder.pt under the OperandFoldManifestSubscript procedure

What was Changed
Added an option to the if statement testing the size of the operand to check for size string. If it is found to be size string, scale the subscripting by that size

```
else
      if operandStkLength[operandStkTop-1] = string then
            subscript := subscript * stringSize;
```

Reason
When subscripting, to find the items in memory, you need to know the size of each element. This change allows the program to process arrays with elements of size string

5.9. Add processing of null character after string
Location of Change
In coder.pt under the AcceptInputToken procedure

What was Changed
Changed the processing of a string to expect a null token at the end of the string

```
read (tCode, compundTokenText[i]);
nextTCodeAddress := nextTCodeAddress + compoundTokenLength + 1;
```

Reason
When subscripting, to find the items in memory, you need to know the size of each element. This change allows the program to process arrays with elements of size string

5.10. Add processing of null character after string
Location of Change
In coder.pt under the oOperandPushChar

What was Changed
Changed the operand to match the string type

```
OperandPush (mStatic, compoundTokenValue, undefined, word)
```

Reason
In PT Pascal, this command was used for single byte char, but in Like it was changed to apply to strings, meaning the data type needs to match

6. String Operations

6.1. New String T-code

Location of Change

In coder.ssl under compound t-codes.

What was Changed

Add following new string T-code:

```
tConcatenate
tRepeatString
tSubstring
tLength
tStringEQ
```

Reason

To have t-codes that can be emitted for the new string operations.

6.2. Add Alternatives for String Operations

Location of Change:

In coder.ssl add alternatives to the OperandPushExpression and OperandPushExpressionAssignPopPop rules.

What was Changed:

The following alternatives were added to the to both the OperandPushExpression and OperandPushExpressionAssignPopPop rules:

```
    | tConcatenate:
            @OperandConcatenatePop
        | tRepeatString:
            @OperandRepeatStringPop
        | tSubstring:
            @OperandSubstringPopPop
        | tLength:
            @OperandLength
        | tStringEQ:
            @OperandStringEqualPop
```

Reason

These alternatives were added to handle the new string operations tConcatenate, tRepeatString, tSubstring, tLength and tStringEQ.

6.3. String Length Rule

Location of Change

In coder.ssl added the OperandLength rule.

What was Changed

This following rule was added for the string length operation, the rule uses the trLength trap code (105) that was previously added:

```
OperandLength:
   % save the temp registers in case in use
    @SaveTempRegsToStack

    % send the argument
    @OperandForceAddressIntoTemp
    @OperandForceToStack
    @OperandPopAndFreeTemp

    % call trLength trap routine
    oOperandPushMode(mTrap)
    oOperandSetValue(trLength)
    oOperandSetLength(word)
    oEmitSingle(iCall)
    oOperandPop

    % pop argument
    oOperandPushMode(mStackReg)
    oOperandSetLength(word)
    oOperandPushMode(mManifest)
    oOperandSetLength(word)
    oOperandSetValue(four)
    oEmitDouble(iAdd)
    oOperandPop
    oOperandPop

    % Accept result
    oOperandPushMode(mScratchReg1)
    oOperandSetLength(word)
    oOperandPushMode(mResultReg)
    oOperandSetLength(word)
    oEmitDouble(iMov)           %       movl    %eax, %S1
    oOperandPop %eax

    % Restore temp regs
    @RestoreTempRegsFromStack   %       popl    %edx .. $eax

    @OperandForceIntoTemp       %       movl    %S1, %T
    oOperandSetLength(byte);
```

Reason

This was added for if a string length operator. The corresponding trap code (trLength) is returned. It takes a string and returns its integer length in a temporary register.

6.4. String Concatenate

Location of Change

In coder.ssl added the OperandConcatenatePop rule.

What was Changed

This following rule was added for the string concatenate operation, the rule uses the trConcatenate trap code (103) that was previously added:

```
OperandConcatenatePop:

    % save the temp registers in case in use
    @SaveTempRegsToStack

    % send the arguments
    @OperandForceToStack
    oOperandPop

    @OperandForceAddressIntoTemp % do this before we can put it into a stack
    @OperandForceToStack
    @OperandPopAndFreeTemp

    % call trConcatenate trap routine
    oOperandPushMode(mTrap)
    oOperandSetValue(trConcatenate)
    oOperandSetLength(word)
    oEmitSingle(iCall)
    oOperandPop

    % pop argument
    oOperandPushMode(mStackReg)
    oOperandSetLength(word)
    oOperandPushMode(mManifest)
    oOperandSetLength(word)
    oOperandSetValue(eight)
    oEmitDouble(iAdd)
    oOperandPop
    oOperandPop

    % Accept result
    oOperandPushMode(mScratchReg1)
    oOperandSetLength(word)
    oOperandPushMode(mResultReg)
    oOperandSetLength(word)
    oEmitDouble(iMov)           %       movl    %eax, %S1
    oOperandPop %eax

    % Restore temp regs
    @RestoreTempRegsFromStack   %       popl    %edx .. $eax

    @OperandForceIntoTemp       %       movl    %S1, %T
    oOperandSetLength(byte);
```

Reason

If a string concatenate operator (i.e. '|') is used then this option will be called. It will emit the concatenate trap. It takes two strings and leaves the address of the string result of their concatenation in a temporary register.

6.5. String Repeat

Location of Change

In coder.ssl added the OperandRepeatStringPop rule.

What was Changed

This following rule was added for the string repeat operation, the rule uses the trRepeat trap code (110) that was previously added:

```
OperandRepeatStringPop:
    % save the temp registers in case in use
    @SaveTempRegsToStack

    % send the arguments
    @OperandForceToStack
    oOperandPop

    @OperandForceAddressIntoTemp % do this before we can put it into a stack
    @OperandForceToStack
    @OperandPopAndFreeTemp

    % call trRepeatString trap routine
    oOperandPushMode(mTrap)
    oOperandSetValue(trRepeatString)
    oOperandSetLength(word)
    oEmitSingle(iCall)
    oOperandPop

    % pop argument
    oOperandPushMode(mStackReg)
    oOperandSetLength(word)
    oOperandPushMode(mManifest)
    oOperandSetLength(word)
    oOperandSetValue(eight)
    oEmitDouble(iAdd)
    oOperandPop
    oOperandPop

    % Accept result
    oOperandPushMode(mScratchReg1)
    oOperandSetLength(word)
    oOperandPushMode(mResultReg)
    oOperandSetLength(word)
    oEmitDouble(iMov)           %       movl    %eax, %S1
    oOperandPop %eax

    % Restore temp regs
    @RestoreTempRegsFromStack   %       popl    %edx .. $eax

    @OperandForceIntoTemp       %       movl    %S1, %T
    oOperandSetLength(byte);
```

Reason

If a string repeat operator (i.e. '||') is used then this option will be called. It will emit the repeat trap. It takes a string, and repetition count and returns the address of the string result in a temporary register.

6.6. Substring

Location of Change

In coder.ssl added the OperandSubstringPopPop rule.

What was Changed

This following rule was added for the string substring operation, the rule using the trSubstring trap code (104) that was previously added:

```
OperandSubstringPopPop:
   % save the temp registers in case in use
    @SaveTempRegsToStack

    % send the arguments
    % Upper index
    @OperandForceToStack
    oOperandPop

    %Lower index
    @OperandForceToStack
    oOperandPop

    %String
    @OperandForceAddressIntoTemp
    @OperandForceToStack
    @OperandPopAndFreeTemp

    % call trSubstring trap routine
    oOperandPushMode(mTrap)
    oOperandSetValue(trSubstring)
    oOperandSetLength(word)
    oEmitSingle(iCall)
    oOperandPop

    % pop argument
    oOperandPushMode(mStackReg)
    oOperandSetLength(word)
    oOperandPushMode(mManifest)
    oOperandSetLength(word)
    oOperandSetValue(twelve)
    oEmitDouble(iAdd)
    oOperandPop
    oOperandPop

    % Accept result
    oOperandPushMode(mScratchReg1)
    oOperandSetLength(word)
    oOperandPushMode(mResultReg)
    oOperandSetLength(word)
```

```
    oEmitDouble(iMov)           %       movl    %eax, %S1
    oOperandPop %eax

    % Restore temp regs
    @RestoreTempRegsFromStack   %       popl    %edx .. $eax

    @OperandForceIntoTemp       %       movl    %S1, %T
    oOperandSetLength(byte);
```

Reason

If a substring operator (e.x. ‘S/1:4’) is used then this option will be called. It will emit the substring trap. It takes a string, a lower index and an upper index and leaves the address of the substring result in a temporary register.

6.7. String Equality

Location of Change

In coder.ssl added the OperandStringEqualPop rule.

What was Changed

This following rule was added for the string equal operation, the rule uses the trStringEqual trap code (105) that was previously added:

```
OperandStringEqualPop:
    % save the temp registers in case in use
    @SaveTempRegsToStack

    % send the arguments
    @OperandForceToStack
    oOperandPop

    @OperandForceAddressIntoTemp % do this before we can put it into a stack
    @OperandForceToStack
    @OperandPopAndFreeTemp

    % call trStringEqual trap routine
    oOperandPushMode(mTrap)
    oOperandSetValue(trStringEqual)
    oOperandSetLength(word)
    oEmitSingle(iCall)
    oOperandPop

    % pop argument
    oOperandPushMode(mStackReg)
    oOperandSetLength(word)
    oOperandPushMode(mManifest)
    oOperandSetLength(word)
    oOperandSetValue(eight)
    oEmitDouble(iAdd)
    oOperandPop
```

```
    oOperandPop

    % Accept result
    oOperandPushMode(mScratchReg1)
    oOperandSetLength(word)
    oOperandPushMode(mResultReg)
    oOperandSetLength(word)
    oEmitDouble(iMov)           %       movl    %eax, %S1
    oOperandPop %eax

    % Restore temp regs
    @RestoreTempRegsFromStack   %       popl    %edx .. $eax

    @OperandForceIntoTemp       %       movl    %S1, %T
    oOperandSetLength(byte);
```

Reason

Added to create code for checking of string equality. It will emit the equality trap. It takes two strings and returns a boolean result in a temporary register.

6.8. Update OperandChr Rule

Location of Change

In coder.ssl added the OperandChar rule.

What was Changed

The OperandChr rule was updated to for strings to the following. Note that the trChrString is now emitted (trap code 102).

```
OperandChr:
  % save the temp registers in case in use
    @SaveTempRegsToStack

    % send the arguments
    % integer
    @OperandForceToStack
    oOperandPop

    % call trConcatenate trap routine
    oOperandPushMode(mTrap)
    oOperandSetValue(trChrString)
    oOperandSetLength(word)
    oEmitSingle(iCall)
    oOperandPop

    % pop argument
    oOperandPushMode(mStackReg)
    oOperandSetLength(word)
    oOperandPushMode(mManifest)
    oOperandSetLength(word)
    oOperandSetValue(four)
    oEmitDouble(iAdd)
```

```
    oOperandPop
    oOperandPop

    % Accept result
    oOperandPushMode(mScratchReg1)
    oOperandSetLength(word)
    oOperandPushMode(mResultReg)
    oOperandSetLength(word)
    oEmitDouble(iMov)           %       movl    %eax, %S1
    oOperandPop %eax

    % Restore temp regs
    @RestoreTempRegsFromStack   %       popl    %edx .. $eax

    @OperandForceIntoTemp       %       movl    %S1, %T
    oOperandSetLength(byte);
```

Reason

Added to take an integer and convert its value to a character and returns the address of the string result in a temporary register.

6.8. Update OperandOrd Rule

Location of Change

In coder.ssl added the OperandOrd rule.

What was Changed

The OperandOrd rule was updated to for strings to the following:

```
OperandOrd:
    %Force address of the string into a temporary.
    @OperandForceAddressIntoTemp % do this before we can put it into a stack
    @OperandForceToStack
    @OperandPopAndFreeTemp

    % Accept result
    % Change the mode to mTempIndirect
    oOperandPushMode(mTempIndirect)
    % change length to byte
    oOperandSetLength(byte)
    oOperandPushMode(mResultReg)
    % change length to byte
    oOperandSetLength(byte)
    oEmitDouble(iMov)           %       movl    %eax, %S1
    oOperandPop %eax

    %force the byte to a temporary to get the result integer (word)
    @OperandForceIntoTemp       %       movb    %S1, %T
    oOperandSetLength(word)
```

Reason
Added to take a string and convert the first character of the string into an integer in a temporary register.

Test Cases

These test cases are located in the source code in the ptsrc/testSuite/ directory. For each test case there are three files:

- The Like test source file (.pt)
- The ssltrace output using -e flag (.eOutput)
- Test documentation (.txt)

The following is a list of our test case files broken down by category. Please see the documentation in the testSuite folder for more information on what each test case covers and a full list of the tests.

Category	**Test Cases**
Empty Program	emptyProgram
Block	blockRule
Strings	stringAssign stringConcatenate stringEquals stringGet stringLength stringNotEqual stringOperationsVarAssignment stringPut stringRepeat stringSubstring stringVal stringVar
Statement	stmtAssignment stmtChooseElse stmtChooseNoElse stmtIfElse stmtRepeatWhile stmtRepeatUnitl
Variable and Types	declarationArray declarationArrayIdentifierBound declarationArrayIdentifierConstBound declarationFileLikeInt declarationFileLikeOutput declarationFileofFiles declarationVarBoolean declarationVarConstant

	declarationVarIdentifieirInt declarationVarIdentifieirString declarationVarInt declarationVarNegInt declarationVarString
Procedure Parameter Types	procedureParameterBoolean procedureParameterIdentifier procedureParameterInt procedureParameterMismatch procedureParameterMulti procedureParameterNone (...)
Initial Values	initialAssignmentConst initialAssignmentExpression initialAssignmentIdentifier initialAssignmentInt initialAssignmentMultiConst initialAssignmentNegInt initialAssignmentPublicConst initialAssignmentPublicVar (...)
Packages	pkgEmpty pkgFun pkgPublicFun pkgPublicVal pkgPublicVar pkgVar

Auto Run Script

Inside the ptsrc directory we have also created a script called runTestsPhase4.sh. This script can be used to autorun all of our test cases, it compiles the code and creates a .s file (these .s file replaces any existing .s files). It compares the output of the ssltrace (using the -e flag) for each file with the .s file and prints the results of which tests were passed. To run this script use the command ./runTestsPhase4.sh in the ptsrc folder.