CSE 101 HW #6

Chris Aikman
Hugo Rivera

April 14, 2015

1 Programming Paradigm Assignment

1.1 Imperative

Example Language:

Java

Application:

Java has been used in thousands of embedded devices, but it may be most commonly known these days for near exclusivity of programming on Android devices. Almost all applications on the Android mobile OS have been programmed with Java.

Android SDK: `https://developer.android.com/sdk/index.html`

Application Appropriateness:

Java was an appropriate choice for designing embedded systems as well as applications for the Android OS because of the large support for the language, as well as its ability to be easily written and read. Java focuses on how states are changed in a program which makes it reliable when debugging and also makes the code easy to read. Code in Java is designed to be modular with emphasis placed on code structure so that programmers joining a project can quickly look through the code and get an understanding of how it works.

1.2 Functional

Example Language:

Haskell

Application:

Haskell has been used to program several web servers, most notably Snap and Yesod.

Snap: `http://snapframework.com/`

Yesod: `http://www.yesodweb.com/`

Application Appropriateness:

Haskell was an appropriate choice for designing these applications because it offers a powerful type system that protects the programmer from making trivial mistakes while still being flexible enough for real world use.

1.3 Object Oriented

Example Language:

C++

Application:

C++ is well known for being the language chosen to implement the two most common web browsers: Google Chrome and Mozilla Firefox.
Google Chrome: `http://www.google.com/chrome/`
Firefox: `https://www.mozilla.org/en-US/firefox/new/`

Application Appropriateness:

C++ was an appropriate choice for these web browsers because it gives the programmer a lot of power while offering the advantages of object oriented programming, which allows code to be smaller, more modular and easier to manage. C++ has also been used, tested and evolved for decades making it extremely reliable when used correctly.

1.4 Logic

Example Language:

Prolog

Application:

Prolog has been successfully used for many years in the creation of artificial intelligence simulations. Most notably, NASA used Prolog to create a voice-operated procedure browser for astronauts.
Programming AI:`http://www.pearsoned.co.uk/highereducation/resources/bratkoprologprogrammingforartificialintelligence3e/`
NASA's Clarissa: `http://ti.arc.nasa.gov/tech/cas/user-centered-technologies/clarissa/`

Application Appropriateness:

Prolog was an appropriate choice for designing artificial intelligence simulations because its focus is on using rules and facts to generate solutions to problems. The logic programming paradigm that Prolog follows makes solving complicated problems that are common to artificial intelligence much easier than thinking of how to solve the issues using a different approach.

2 Parallel Matrix Multiply Assignment

2.1 Pseudocode

Haskell is a declarative language, thus this is declarative pseudocode.

```
a matrix is a 2D array of elements that can be multiplied and added

to multiply two matrices a (dimensions n by m), b (dimensions m by p):
    let the result be matrix c of dimensions
    let m = shared dimension of b and a
            if there is no shared dimension, throw an error

    map (ra ->
        map (cb ->
            sum [ra[i]*b[i] | i<-[0..m]])
        b.cols)
    a.rows
```

2.2 Haskell Code

```haskell
module MatrixMult (multMatrix) where
import Data.List (tranpose)

multMatrix a b =
    map (\ra ->
        map (\cb ->
            sum $ zipWith (*) ra cb) (transpose b)) a
```

2.3 Assembly Code Computation

Done using "ghc -S matrix_mult.hs" See the appendix.

2.4 Assembly Code Multiplication

This happens in line 52 after preparing the arguments. Haskell performs a jump the number library's integer multiplication method "jmp base_GHCziNum_zt_info" (instructions for Base.GHC.zahltimes or integer mult)

2.5 Number of Assembly Code Lines for Multiplication

It takes 6 lines of code starting at line 47 to gather the arguments and prepare the program for jumping to "base_GHCziNum_zt_info"

3 Appendix

```
.data
        .align 8
.align 1
.globl __stginit_MatrixMult
.type __stginit_MatrixMult, @object
__stginit_MatrixMult:
.data
        .align 8
.align 1
.globl MatrixMult_multMatrix_closure
.type MatrixMult_multMatrix_closure, @object
MatrixMult_multMatrix_closure:
        .quad   MatrixMult_multMatrix_info
        .quad   0
.text
        .align 8
        .long   SSg_srt-(sRo_info)+0
        .long   0
        .quad   1
        .quad   4294967313
sRo_info:
.LcRD:
        leaq -16(%rbp),%rax
        cmpq %r15,%rax
        jb .LcRE
.LcRF:
        movq $stg_upd_frame_info,-16(%rbp)
        movq %rbx,-8(%rbp)
        movq 16(%rbx),%rax
        movq %rax,%r14
        movl $base_DataziList_transpose_closure,%ebx
        addq $-16,%rbp
        jmp stg_ap_p_fast
.LcRE:
        jmp *-16(%r13)
        .size sRo_info, .-sRo_info
.text
        .align 8
```

```
	.quad	1
	.quad	17
sRl_info:
.LcRT:
	leaq -16(%rbp),%rax
	cmpq %r15,%rax
	jb .LcRU
.LcRV:
	movq $stg_upd_frame_info,-16(%rbp)
	movq %rbx,-8(%rbp)
	movq 16(%rbx),%rax
	movq %rax,%r14
	addq $-16,%rbp
	jmp base_GHCziNum_zt_info
.LcRU:
	jmp *-16(%r13)
	.size sRl_info, .-sRl_info
.text
	.align 8
	.long	SSg_srt-(sRm_info)+8
	.long	0
	.quad	3
	.quad	4294967312
sRm_info:
.LcRW:
	leaq -16(%rbp),%rax
	cmpq %r15,%rax
	jb .LcRX
.LcRY:
	addq $24,%r12
	cmpq 856(%r13),%r12
	ja .LcS0
.LcRZ:
	movq $stg_upd_frame_info,-16(%rbp)
	movq %rbx,-8(%rbp)
	movq 16(%rbx),%rax
	movq 24(%rbx),%rcx
	movq 32(%rbx),%rbx
	movq $sRl_info,-16(%r12)
	movq %rax,(%r12)
	leaq -16(%r12),%rax
```

```
        movq %rbx,%rdi
        movq %rcx,%rsi
        movq %rax,%r14
        movl $base_GHCziList_zzipWith_closure,%ebx
        addq $-16,%rbp
        jmp stg_ap_ppp_fast
.LcS0:
        movq $24,904(%r13)
.LcRX:
        jmp *-16(%r13)
        .size sRm_info, .-sRm_info
.text
        .align 8
        .long   SSg_srt-(sRn_info)+8
        .long   0
        .quad   4294967301
        .quad   2
        .quad   12884901900
sRn_info:
.LcS1:
.LcS3:
        addq $40,%r12
        cmpq 856(%r13),%r12
        ja .LcS5
.LcS4:
        movq 7(%rbx),%rax
        movq 15(%rbx),%rbx
        movq $sRm_info,-32(%r12)
        movq %rax,-16(%r12)
        movq %rbx,-8(%r12)
        movq %r14,(%r12)
        leaq -32(%r12),%rbx
        movq %rbx,%rsi
        movq %rax,%r14
        movl $base_DataziList_sum_closure,%ebx
        jmp stg_ap_pp_fast
.LcS5:
        movq $40,904(%r13)
.LcS2:
        jmp *-8(%r13)
        .size sRn_info, .-sRn_info
```

```
.text
        .align 8
        .long   SSg_srt-(sRp_info)+0
        .long   0
        .quad   4294967301
        .quad   2
        .quad   64424509452
sRp_info:
.LcS6:
.LcS8:
        addq $48,%r12
        cmpq 856(%r13),%r12
        ja .LcSa
.LcS9:
        movq 7(%rbx),%rax
        movq 15(%rbx),%rbx
        movq $sRo_info,-40(%r12)
        movq %rbx,-24(%r12)
        leaq -40(%r12),%rbx
        movq $sRn_info,-16(%r12)
        movq %rax,-8(%r12)
        movq %r14,(%r12)
        leaq -15(%r12),%rax
        movq %rbx,%rsi
        movq %rax,%r14
        movl $base_GHCziBase_map_closure,%ebx
        jmp stg_ap_pp_fast
.LcSa:
        movq $48,904(%r13)
.LcS7:
        jmp *-8(%r13)
        .size sRp_info, .-sRp_info
.text
        .align 8
        .long   SSg_srt-(MatrixMult_multMatrix_info)+0
        .long   0
        .quad   12884901911
        .quad   0
        .quad   133143986191
.globl MatrixMult_multMatrix_info
.type MatrixMult_multMatrix_info, @object
```

```
MatrixMult_multMatrix_info:
.LcSb:
.LcSd:
        addq $24,%r12
        cmpq 856(%r13),%r12
        ja .LcSf
.LcSe:
        movq $sRp_info,-16(%r12)
        movq %r14,-8(%r12)
        movq %rdi,(%r12)
        leaq -15(%r12),%rax
        movq %rax,%r14
        movl $base_GHCziBase_map_closure,%ebx
        jmp stg_ap_pp_fast
.LcSf:
        movq $24,904(%r13)
.LcSc:
        movl $MatrixMult_multMatrix_closure,%ebx
        jmp *-8(%r13)
        .size MatrixMult_multMatrix_info, .-MatrixMult_multMatrix_info
.section .data
        .align 8
.align 1
SSg_srt:
        .quad   base_DataziList_transpose_closure
        .quad   base_GHCziList_zzipWith_closure
        .quad   base_DataziList_sum_closure
        .quad   base_GHCziBase_map_closure
        .quad   MatrixMult_multMatrix_closure
.section .note.GNU-stack,"",@progbits
.ident "GHC 7.8.3"
```