Component Based MMIX Simulator using Multiple Programming Paradigms

A dissertation submitted in partial fulfilment of the requirements for the MSc in Advanced Computing Technologies

by Stephen Edmans

Department of Computer Science and Information Systems
Birkbeck College, University of London

September 2015

This report is substantially the result of my own work except where explicitly indicated in the text. I give my permission for it to be submitted to the JISC Plagiarism Detection Service. I have read and understood the sections on plagiarism in the Programme Handbook and the College website.

Abstract

There are currently over 2,500[1] different programming languages, with more created every year. These programming languages can get grouped together in numerous different ways. This makes the decision of what language to use when starting a new project extremely difficult.

There are several ways in which we can reach this decision; choose the language that your team knows best; choose the language that makes the most sense to implement the critical part of your system; choose a simple general purpose language; choose a language that has got an active community. There is no acknowledged best approach to take.

Another approach would be to split your application up into separate components and using a different programming language for each component. This allows us choose the most appropriate programming language for each component.

The purpose of this project is to examine this approach, the application that we will create will be a simulator for an artificial machine language. The artificial machine language that we will use is called MMIX, it was developed by Donald Knuth as part of his seminal work The Art of Computer Programming[Knu11].

[1] From the language list[Kin]

Contents

List of Figures

Acknowledgements

Chapter 1

Introduction

Chapter 2

Assembler

2.1 Introduction

2.2 Executable

2.3 Lexer

2.4 Parser

2.5 Code Generation

2.5.1 Symbol Table

2.5.2 Automatically Assigned Registers

2.5.3 Local Symbols

2.5.4 Handling Operands

2.5.5 Assembler Directives

2.5.6 Generating the Output

2.6 Component Testing

Chapter 3

Graphical User Interface

3.1 Introduction

3.2 User Interface Design

3.2.1 Console Panel

3.2.2 Controls Panel

3.2.3 Main State Panel

3.2.4 Memory Panel

3.2.5 Registers Panel

3.3 Asynchronous User Interface Programming with Actors

3.4 Communication

3.5 Component Testing

Chapter 4

Virtual Machine

Chapter 5

Simulator Application

5.1 Introduction

5.2 Integration Testing

5.2.1 Generate Prime Numbers Sample Application

Conclusion

References

[akk] Akka toolkit. <http://akka.io/ >[Access 24 August 2015].

[Kin] Bill Kinnersley. The language list. <http://people.ku.edu/~nkinners/LangList/Extras/langlist.htm >[Access 7 September 2015].

[Knu] D.E. Knuth. The art of computer programming fascicle 1 mmix [e-book]. Stanford University: Addison Wesley Available through: Stanford University <http://www-cs-faculty.stanford.edu/~uno/fasc1.ps.gz>[Access 7 April 2013].

[Knu90] D.E. Knuth. *MMIXware A RISC Computer for the Third Millennium*. Springer, 1990.

[Knu11] D.E. Knuth. *The Art of Computer Programming*, volume 1-4a. 1st ed. Addison Wesley, 2011.

[Ruc12] Martin Ruckert. Mmix quick reference card. <http://mmix.cs.hm.edu/doc/mmix-refcard-a4.pdf >[Access 24 August 2015], 2012.

Appendix A

Source Code

A.1 Assembler

A.2 Graphical User Interface

A.3 Virtual Machine

Appendix B

Intermediate Assembler Representations

B.1 Definitions

B.2 Test Application

B.2.1 Sample Test MMIXAL Code

The sample mmixal application I am using to test the system is contained in Fascile 1[Knu]. The complete code listing is

```
L       IS    500
t       IS    $255
n       GREG  0
q       GREG  0
r       GREG  0
jj      GREG  0
kk      GREG  0
pk      GREG  0
mm      IS    kk
        LOC   Data_Segment
PRIME1  WYDE  2
        LOC   PRIME1+2*L
ptop    GREG  @
j0      GREG  PRIME1+2-@
BUF     OCTA  0
        LOC   #100
Main    SET   n,3
        SET   jj,j0
2H      STWU  n,ptop,jj
        INCL  jj,2
3H      BZ    jj,2F
4H      INCL  n,2
5H      SET   kk,j0
6H      LDWU  pk,ptop,kk
        DIV   q,n,pk
        GET   r,rR
        BZ    r,4B
7H      CMP   t,q,pk
        BNP   t,2B
8H      INCL  kk,2
        JMP   6B
        GREG  @
Title   BYTE  "First Five Hundred Primes"
NewLn   BYTE  #a,0
Blanks  BYTE  "   ",0
2H      LDA   t,Title
        TRAP  0,Fputs,StdOut
        NEG   mm,2
3H      ADD   mm,mm,j0
        LDA   t,Blanks
        TRAP  0,Fputs,StdOut
2H      LDWU  pk,ptop,mm
0H      GREG  #2030303030000000
        STOU  0B,BUF
        LDA   t,BUF+4
1H      DIV   pk,pk,10
        GET   r,rR
        INCL  r,'0'
        STBU  r,t,0
        SUB   t,t,1
        PBNZ  pk,1B
        LDA   t,BUF
        TRAP  0,Fputs,StdOut
        INCL  mm,2*L/10
        PBN   mm,2B
        LDA   t,NewLn
        TRAP  0,Fputs,StdOut
        CMP   t,mm,2*(L/10-1)
        PBNZ  t,3B
        TRAP  0,Halt,0
```

B.2.2 Parsed Sample File

The final version of the parsed source code for the test application is

```
LabelledPILine {
        lppl_id = IsNumber 500, lppl_ident = Id "L", lppl_loc = 0
}
LabelledPILine {
        lppl_id = IsRegister 255, lppl_ident = Id "t", lppl_loc = 0
}
LabelledPILine {
        lppl_id = GregEx (ExpressionRegister '\254' (ExpressionNumber
            0)), lppl_ident = Id "n", lppl_loc = 0
}
LabelledPILine {
        lppl_id = GregEx (ExpressionRegister '\253' (ExpressionNumber
            0)), lppl_ident = Id "q", lppl_loc = 0
}
LabelledPILine {
        lppl_id = GregEx (ExpressionRegister '\252' (ExpressionNumber
            0)), lppl_ident = Id "r", lppl_loc = 0
}
LabelledPILine {
        lppl_id = GregEx (ExpressionRegister '\251' (ExpressionNumber
            0)), lppl_ident = Id "jj", lppl_loc = 0
}
LabelledPILine {
        lppl_id = GregEx (ExpressionRegister '\250' (ExpressionNumber
            0)), lppl_ident = Id "kk", lppl_loc = 0
}
LabelledPILine {
        lppl_id = GregEx (ExpressionRegister '\249' (ExpressionNumber
            0)), lppl_ident = Id "pk", lppl_loc = 0
}
LabelledPILine {
        lppl_id = GregEx (ExpressionRegister '\248' (ExpressionNumber
            0)), lppl_ident = Id "mm", lppl_loc = 0
}
PlainPILine {
        ppl_id = LocEx (ExpressionNumber 536870912), ppl_loc =
            536870912
}
LabelledPILine {
        lppl_id = WydeArray "\STX", lppl_ident = Id "PRIME1", lppl_loc
             = 536870912
}
PlainPILine {
        ppl_id = LocEx (ExpressionNumber 536871912), ppl_loc =
            536871912
}
LabelledPILine {
        lppl_id = GregEx (ExpressionRegister '\247' (ExpressionNumber
            536871912)), lppl_ident = Id "ptop", lppl_loc = 536871912
}
LabelledPILine {
        lppl_id = GregEx (ExpressionRegister '\246' (ExpressionNumber
            (-998))), lppl_ident = Id "j0", lppl_loc = 536871912
}
LabelledPILine {
        lppl_id = OctaArray "\NUL", lppl_ident = Id "BUF", lppl_loc =
            536871912
}
```

```
PlainPILine {
        ppl_id = LocEx (ExpressionNumber 256), ppl_loc = 256
}
LabelledPILine {
        lppl_id = Set (Expr (ExpressionIdentifier (Id "n")),Expr (
            ExpressionNumber 3)), lppl_ident = Id "Main", lppl_loc =
            256
}
PlainPILine {
        ppl_id = Set (Expr (ExpressionIdentifier (Id "jj")),Expr (
            ExpressionIdentifier (Id "j0"))), ppl_loc = 260
}
LabelledOpCodeLine {
        lpocl_code = 166, lpocl_ops = [Expr (ExpressionIdentifier (Id
            "n")),Expr (ExpressionIdentifier (Id "ptop")),Expr (
            ExpressionIdentifier (Id "jj"))], lpocl_ident = Id "??2H0"
            , lpocl_loc = 264
}
PlainOpCodeLine {
        pocl_code = 231, pocl_ops = [Expr (ExpressionIdentifier (Id "
            jj")),Expr (ExpressionNumber 2)], pocl_loc = 268
}
LabelledOpCodeLine {
        lpocl_code = 66, lpocl_ops = [Expr (ExpressionIdentifier (Id "
            jj")),Ident (Id "??2H1")], lpocl_ident = Id "??3H0",
            lpocl_loc = 272
}
LabelledOpCodeLine {
        lpocl_code = 231, lpocl_ops = [Expr (ExpressionIdentifier (Id
            "n")),Expr (ExpressionNumber 2)], lpocl_ident = Id "??4H0"
            , lpocl_loc = 276
}
LabelledPILine {
        lppl_id = Set (Expr (ExpressionIdentifier (Id "kk")),Expr (
            ExpressionIdentifier (Id "j0"))), lppl_ident = Id "??5H0",
             lppl_loc = 280
}
LabelledOpCodeLine {
        lpocl_code = 134, lpocl_ops = [Expr (ExpressionIdentifier (Id
            "pk")),Expr (ExpressionIdentifier (Id "ptop")),Expr (
            ExpressionIdentifier (Id "kk"))], lpocl_ident = Id "??6H0"
            , lpocl_loc = 284
}
PlainOpCodeLine {
        pocl_code = 28, pocl_ops = [Expr (ExpressionIdentifier (Id "q"
            )),Expr (ExpressionIdentifier (Id "n")),Expr (
            ExpressionIdentifier (Id "pk"))], pocl_loc = 288
}
PlainOpCodeLine {
        pocl_code = 254, pocl_ops = [Expr (ExpressionIdentifier (Id "r
            ")),Expr (ExpressionIdentifier (Id "rR"))], pocl_loc = 292
}
PlainOpCodeLine {
        pocl_code = 66, pocl_ops = [Expr (ExpressionIdentifier (Id "r"
            )),Ident (Id "??4H0")], pocl_loc = 296
}
LabelledOpCodeLine {
        lpocl_code = 48, lpocl_ops = [Expr (ExpressionIdentifier (Id "
            t")),Expr (ExpressionIdentifier (Id "q")),Expr (
            ExpressionIdentifier (Id "pk"))], lpocl_ident = Id "??7H0"
            , lpocl_loc = 300
}
```

```
PlainOpCodeLine {
        pocl_code = 76, pocl_ops = [Expr (ExpressionIdentifier (Id "t"
            )),Ident (Id "??2H0")], pocl_loc = 304
}
LabelledOpCodeLine {
        lpocl_code = 231, lpocl_ops = [Expr (ExpressionIdentifier (Id
            "kk")),Expr (ExpressionNumber 2)], lpocl_ident = Id "??8H0
            ", lpocl_loc = 308
}
PlainOpCodeLine {
        pocl_code = 240, pocl_ops = [Ident (Id "??6H0")], pocl_loc =
            312
}
PlainPILine {
        ppl_id = GregEx (ExpressionRegister '\245' ExpressionAT),
            ppl_loc = 316
}
LabelledPILine {
        lppl_id = ByteArray "First␣Five␣Hundred␣Primes", lppl_ident =
            Id "Title", lppl_loc = 316
}
LabelledPILine {
        lppl_id = ByteArray "\n\NUL", lppl_ident = Id "NewLn",
            lppl_loc = 341
}
LabelledPILine {
        lppl_id = ByteArray "␣␣␣\NUL", lppl_ident = Id "Blanks",
            lppl_loc = 343
}
LabelledOpCodeLine {
        lpocl_code = 34, lpocl_ops = [Expr (ExpressionIdentifier (Id "
            t")),Expr (ExpressionIdentifier (Id "Title"))],
            lpocl_ident = Id "??2H1", lpocl_loc = 347
}
PlainOpCodeLine {
        pocl_code = 0, pocl_ops = [Expr (ExpressionNumber 0),
            PseudoCode 7,PseudoCode 1], pocl_loc = 351
}
PlainOpCodeLine {
        pocl_code = 52, pocl_ops = [Expr (ExpressionIdentifier (Id "mm
            ")),Expr (ExpressionNumber 2)], pocl_loc = 355
}
LabelledOpCodeLine {
        lpocl_code = 32, lpocl_ops = [Expr (ExpressionIdentifier (Id "
            mm")),Expr (ExpressionIdentifier (Id "mm")),Expr (
            ExpressionIdentifier (Id "j0"))], lpocl_ident = Id "??3H1"
            , lpocl_loc = 359
}
PlainOpCodeLine {
        pocl_code = 34, pocl_ops = [Expr (ExpressionIdentifier (Id "t"
            )),Expr (ExpressionIdentifier (Id "Blanks"))], pocl_loc =
            363
}
PlainOpCodeLine {
        pocl_code = 0, pocl_ops = [Expr (ExpressionNumber 0),
            PseudoCode 7,PseudoCode 1], pocl_loc = 367
}
LabelledOpCodeLine {
        lpocl_code = 134, lpocl_ops = [Expr (ExpressionIdentifier (Id
            "pk")),Expr (ExpressionIdentifier (Id "ptop")),Expr (
            ExpressionIdentifier (Id "mm"))], lpocl_ident = Id "??2H2"
            , lpocl_loc = 371
```

```
}
LabelledPILine {
        lppl_id = GregEx (ExpressionRegister '\244' (ExpressionNumber
            2319406791617675264)), lppl_ident = Id "??0H0", lppl_loc =
             375
}
PlainOpCodeLine {
        pocl_code = 174, pocl_ops = [Ident (Id "??0H0"),Expr (
            ExpressionIdentifier (Id "BUF"))], pocl_loc = 375
}
PlainOpCodeLine {
        pocl_code = 34, pocl_ops = [Expr (ExpressionIdentifier (Id "t"
            )),Expr (ExpressionNumber 536870924)], pocl_loc = 379
}
LabelledOpCodeLine {
        lpocl_code = 28, lpocl_ops = [Expr (ExpressionIdentifier (Id "
            pk")),Expr (ExpressionIdentifier (Id "pk")),Expr (
            ExpressionNumber 10)], lpocl_ident = Id "??1H0", lpocl_loc
             = 383
}
PlainOpCodeLine {
        pocl_code = 254, pocl_ops = [Expr (ExpressionIdentifier (Id "r
            ")),Expr (ExpressionIdentifier (Id "rR"))], pocl_loc = 387
}
PlainOpCodeLine {
        pocl_code = 231, pocl_ops = [Expr (ExpressionIdentifier (Id "r
            ")),Expr (ExpressionNumber 48)], pocl_loc = 391
}
PlainOpCodeLine {
        pocl_code = 162, pocl_ops = [Expr (ExpressionIdentifier (Id "r
            ")),Expr (ExpressionIdentifier (Id "t")),Expr (
            ExpressionNumber 0)], pocl_loc = 395
}
PlainOpCodeLine {
        pocl_code = 36, pocl_ops = [Expr (ExpressionIdentifier (Id "t"
            )),Expr (ExpressionIdentifier (Id "t")),Expr (
            ExpressionNumber 1)], pocl_loc = 399
}
PlainOpCodeLine {
        pocl_code = 90, pocl_ops = [Expr (ExpressionIdentifier (Id "pk
            ")),Ident (Id "??1H0")], pocl_loc = 403
}
PlainOpCodeLine {
        pocl_code = 34, pocl_ops = [Expr (ExpressionIdentifier (Id "t"
            )),Expr (ExpressionIdentifier (Id "BUF"))], pocl_loc = 407
}
PlainOpCodeLine {
        pocl_code = 0, pocl_ops = [Expr (ExpressionNumber 0),
            PseudoCode 7,PseudoCode 1], pocl_loc = 411
}
PlainOpCodeLine {
        pocl_code = 231, pocl_ops = [Expr (ExpressionIdentifier (Id "
            mm")),Expr (ExpressionNumber 100)], pocl_loc = 415
}
PlainOpCodeLine {
        pocl_code = 80, pocl_ops = [Expr (ExpressionIdentifier (Id "mm
            ")),Ident (Id "??2H2")], pocl_loc = 419
}
PlainOpCodeLine {
        pocl_code = 34, pocl_ops = [Expr (ExpressionIdentifier (Id "t"
            )),Expr (ExpressionIdentifier (Id "NewLn"))], pocl_loc =
            423
```

```
}
PlainOpCodeLine {
        pocl_code = 0, pocl_ops = [Expr (ExpressionNumber 0),
            PseudoCode 7,PseudoCode 1], pocl_loc = 427
}
PlainOpCodeLine {
        pocl_code = 48, pocl_ops = [Expr (ExpressionIdentifier (Id "t"
            )),Expr (ExpressionIdentifier (Id "mm")),Expr (
            ExpressionNumber 98)], pocl_loc = 431
}
PlainOpCodeLine {
        pocl_code = 90, pocl_ops = [Expr (ExpressionIdentifier (Id "t"
            )),Ident (Id "??3H1")], pocl_loc = 435
}
PlainOpCodeLine {
        pocl_code = 0, pocl_ops = [Expr (ExpressionNumber 0),
            PseudoCode 0,Expr (ExpressionNumber 0)], pocl_loc = 439
}
```