Προαιρετική Εργασία στο μάθημα Σχεδίαση Γλωσσών Προγραμματισμού 2018

Όνομα: Γεώργιος - Δαυίδ Τσεκάλας
ΑΕΜ: 2888

Υλοποίηση

Ο μεταγλωττιστής έχει γραφτεί σε flex (2.6.4) και GNU bison (3.0.4) και έχει μεταγλωττιστεί με τον gcc (version 8.1.0). Για τη δημιουργια του compiler αρκει η εντολη **make** (version 4.2.1). Στο *Makefile* υπαρχουν οι ρουτίνες ***build*** (απλο build για παραγωγη κωδικα), ***build_test*** (για παραγωγη κωδικα και τυπωμα του συντακτικου δεντρου και του πίνακα συμβόλων), ***test*** (για να τυπωθούν ολα τα test files) και ***clean*** (για τον καθαρισμο των παραγόμενων αρχείων). Αναλυτικότερα η ρουτίνα test απαιτεί το mixvm απο το GNU mdk (mixasm, mixvm) και bash.

Σημείωση: τα version των προγραμμάτων είναι εκείνα στα οποία έκανα εγώ το compile.

Μετά το build χρειαζεται να δωσουμε αυτη την εντολη για την παραγωγη του κωδικα mix

cat your_file_to_compile.source | ./compiler

Μεσα στον φακελο του μεταγλωττιστη υπαρχει ο φακελος tests (που περιεχει μερικα test για compilation τα οποια μπορουν να τυποθουν αυτοματα μεσω της *make test*), τον φακελο src (ο οποίος περιέχει τον κώδικα του προγράμματος) και το bash αρχειο test_mixal.sh που ειναι ουσιαστικά η make test.

Στον φάκελο src βρίσκεται το πρόγραμμα. Το αρχειο lex.l και parse.y ειναι ο lexer και ο parser αντίστοιχα. Η helper_funcs.* περιέχει τις συναρτήσεις για τη δημιουργία του συντακτικού δέντρου και την παραγωγη κώδικα (***eval()***). Στο αχειο symbols.* υπάρχουν οι συναρτήσεις για τη κατασκευή του πίνακα συμβόλων.

Γενικά η κατασκευή του συντακτικού δέντρου ξεκινά από κάτω προς τα πάνω όπως συμβαίνει και στον parser. Κάθε κόμβος δημιουργείται με τη συναρτηση ***newnode()*** ενω οι συναρτήσεις ***newnode_*()*** ειναι wrappers της, για καλύτερη ανάγνωση του κώδικα.

Αφου έχει δημιουργηθεί το st δινουμε το PROGRAM node (κορυφη του δεντρου) στην ***eval()*** και ξεκινά το evaluation με κατεύθυνση κυριως απο τα αριστερά προς τα δεξιά εκτός καποιον περιπτώσεων, για παράδειγμα στην αφαίρεση για τον λόγο ότι μας ενδιαφέρει σειρά των αριθμών.
Το σημαντικότερο μερος του code generation είναι η συναρτηση ***save_or_not_valueof_A()*** γιατι ελεγχει αν θα αποθηκεύσει το αποτελεσμα του A reg στο “stack”. Το σκεπτικό είναι το εξής, η πρώτη μεριά που κανει πρωτα, αναδρομικα evaluate είναι υποψήφια για το “stack”. Όταν κινείται μέσα στην αναδρομή ελεγχει αν ο κομβος ειναι ***IN_TEMP_MEMORY*** (“stack”) και το κανει load απο εκει σε αντίθεση με ενα hardcoded load.

Ο πίνακας συμβόλων ακολουθεί το πρότυπο των commit του git. Ενα hash table με κλειδι το πρωτο γραμμα (στο git είναι δύο γράμματα) της μεταβλητής που καταλήγει σε μια single linked list. Καθε node περιεχει το ονομα της μεταβλητής και τον αριθμό που εχει ως VAR* στο code generation.

Γλώσσα

Αν και ελάχιστα τα παραδείγματα στο web για τη γλωσσα mix, το βιβλιο του Donald Knuth και το υλικό που δόθηκε στην εκφώνηση της εργασίας ήταν αρκετά.

Η mix τρέχει σε ενα vm με 4000 διευθύνσεις μνήμης, στις οποίες είναι ελεύθερος ο προγραμματιστης να οργανωσει το προγραμμα του. Δεν υπάρχουν οι εννοιες των memory segments (πχ. stack, heap, data, code).

Προτιμάται η χρήση του A register για τον λογο οτι οι περισσότερες εντολές τον χρησιμοποιούν αποκλειστικα (πχ ADD, SUB).

Οπως φένεται και στον compiler, στις συναρτησεις ***generate_instr()*** και ***constant_load_big_number()*** υπάρχει το προβλημα οτι δεν γινεται να χωρέσουν όλοι οι αριθμοί σε συγκεκριμένα registers οπότε είναι αναγκαία η εισαγωγή σε μια τοποθεσία στην μνήμη. Κατι το οποιο ειναι φυσιολογικό αφου πρεπει μεσα σε 32 bit να χωρέσει το instruction και ο αριθμος.

Γενικά, η γλώσσα mix βρισκεται κοντα στην σημερινη x86-64 assembly και εκπληρώνει το στοχο της, που ειναι η εκμαθηση, αφου καταφέρνει και κρυβει απο τον προγραμματιστή αρκετα τεχνικα ζητηματα που προκύπτουν απο τα συγχρονα λειτουργικα συστηματα.

Μερικά Παραδείγματα κώδικα - *make test*

Όλα τα παρακάτω προέρχονται από την έξοδο της εκτέλεσης του *make test.*

```
OUTBUFF EQU 51
OUTBUFF1        EQU 52
OUTBUFF2        EQU 53

TEMP    EQU 3500
TEMP2   EQU 3501
STACK   EQU 3502
START   EQU 80
        ORIG   START

VAR1    EQU 1   a
        ENTA 2
        MUL =2=
        JANZ ERROR
        STX TEMP
        LDA TEMP
        DECA 1
        JOV ERROR
        DECA 3
        JOV ERROR
        STA STACK
        JAZ LAND0
        LDA =10000=
        MUL =1337=
        JANZ ERROR
        STX TEMP
        LDA TEMP
LAND0   NOP
        JANZ LOR0
        ENTA 1
        CMPA =0=
        JG *+3
        ENTA 0
        JMP *+2
        ENTA 1
LOR0    NOP
        STA VAR1
        LDA VAR1
        STA TEMP
        ENTX 45
        JAN *+2
        ENTX 44
        STX OUTBUFF
        LDA TEMP
        CHAR
        STA OUTBUFF1
        STX OUTBUFF2
        OUT OUTBUFF(19)
        JBUS *(19)
        ENTA 0
        HLT
ERROR   ENTA 1
        HLT
        END START
```

```
########## logic.source ###############
{
        var a:int;
        a = 2*2-1-3 && 1337*10000 || 1 > 0;
        print a;
}

// all verified with c
a = (0 || 0) || 1 && 0; // output 0
a = (1 || 2-2+1-1) || 1 && 0; // output 1
a = 2*2-1-3 && 1337*10000 || 1 > 0; output 1
a = 1 < 1; // output 0

--------- MIXVM OUTPUT ----------

    +0000000001
rA: + 00 00 00 00 00 (0000000000)
rX: + 30 30 30 30 31 (0511305631)
rJ: + 01 37 (0101)
rI1: + 00 00 (0000)     rI2: + 00 00 (0000)
rI3: + 00 00 (0000)     rI4: + 00 00 (0000)
rI5: + 00 00 (0000)     rI6: + 00 00 (0000)
```

Ο κώδικας της λογικής έκφρασης.

Εκτέλεση του κώδικα που δίνεται στην εκφώνηση της εργασίας. (fibonacci)

```
OUTBUFF EQU 51
OUTBUFF1        EQU 52
OUTBUFF2        EQU 53

TEMP    EQU 3500
TEMP2   EQU 3501
STACK   EQU 3502
START   EQU 80
        ORIG   START

VAR4    EQU 4   first
VAR3    EQU 3   second
VAR2    EQU 2   i
VAR1    EQU 1   tmp
        ENTA 0
        STA VAR4
        ENTA 1
        STA VAR3
        ENTA 0
        STA VAR1
LLOOP0  LDA VAR2
        CMPA =10=
        JL *+3
        ENTA 0
        JMP *+2
        ENTA 1
        JAZ LLOOP1
        ENTA 1
        ADD VAR2
        JOV ERROR
        STA VAR2
        LDA VAR3
        ADD VAR4
        JOV ERROR
        STA VAR1
        LDA VAR1
        STA TEMP
        ENTX 45
        JAN *+2
        ENTX 44
        STX OUTBUFF
        LDA TEMP
        CHAR
        STA OUTBUFF1
        STX OUTBUFF2
        OUT OUTBUFF(19)
        JBUS *(19)
        LDA VAR3
        STA VAR4
        LDA VAR1
        STA VAR3
        JMP LLOOP0
LLOOP1  NOP
        ENTA 0
        HLT
ERROR   ENTA 1
        HLT
        END START
```

```
########### test_pdf.source ###############
{
        var first, second, i, tmp: int;

        first=0;
        second=1;
        tmp=0;
        while(i<10)
        {

                i=i+1;
                tmp=first+second;
                print tmp;
                first=second;
                second=tmp;
        }
}

// put 701408733 first 44 fibonacci numbers



--------- MIXVM OUTPUT ----------

    +0000000001
    +0000000002
    +0000000003
    +0000000005
    +0000000008
    +0000000013
    +0000000021
    +0000000034
    +0000000055
    +0000000089
rA: + 00 00 00 00 00 (0000000000)
rX: + 30 30 30 38 39 (0511306151)
rJ: + 01 29 (0093)
rI1: + 00 00 (0000)     rI2: + 00 00 (0000)
rI3: + 00 00 (0000)     rI4: + 00 00 (0000)
rI5: + 00 00 (0000)     rI6: + 00 00 (0000)
```

Εκτέλεση του κώδικα που δίνεται στην εκφώνηση της εργασίας. (division-multiplication-modulo)

```
VAR2    EQU 2   k
VAR1    EQU 1   a
        ENTA 3
        STA VAR3
        ENTA 2
        STA VAR2
        LDA VAR3
        STA TEMP
        LDX TEMP
        ENTA 0
        DIV VAR2
        JOV ERROR
        STA STACK
        LDA VAR2
        MUL STACK
        JANZ ERROR
        STX TEMP
        LDA TEMP
        STA STACK
        LDA VAR3
        STA TEMP
        LDX TEMP
        ENTA 0
        DIV VAR2
        JOV ERROR
        STX TEMP
        LDA TEMP
        ADD STACK
        JOV ERROR
        STA VAR1
        LDA VAR3
        STA TEMP
        ENTX 45
        JAN *+2
        ENTX 44
        STX OUTBUFF
        LDA TEMP
        CHAR
        STA OUTBUFF1
        STX OUTBUFF2
        OUT OUTBUFF(19)
        JBUS *(19)
        LDA VAR1
        STA TEMP
        ENTX 45
        JAN *+2
        ENTX 44
        STX OUTBUFF
        LDA TEMP
        CHAR
        STA OUTBUFF1
        STX OUTBUFF2
        OUT OUTBUFF(19)
        JBUS *(19)
        ENTA 0
        HLT
ERROR   ENTA 1
        HLT
        END START
```

```
############ math-mul-div-pdf.source ###############
{
        var i,k,a :int;
        i = 3;
        k = 2;
        a = (i / k) * k + (i % k);

        print i;
        print a;
}

// a(1) = i = 3;

--------- MIXVM OUTPUT ----------

    +0000000003
    +0000000003
rA: + 00 00 00 00 00 (0000000000)
rX: + 30 30 30 30 33 (0511305633)
rJ: + 00 00 (0000)
rI1: + 00 00 (0000)     rI2: + 00 00 (0000)
rI3: + 00 00 (0000)     rI4: + 00 00 (0000)
rI5: + 00 00 (0000)     rI6: + 00 00 (0000)
```

(Αριστερά - Ο πρόλογος καθώς και μερικές δηλώσεις μεταβλητών έχουν κοπεί από την εικόνα)