



Linear ACSE

Ettore
Speziale

Introduction

Assignment

Expression

Arithmetic
Comparison

Bibliography

Linear ACSE

Ettore Speziale

Politecnico di Milano



Contents

Linear ACSE

Ettore
Speziale

Introduction

Assignment

Expression
Arithmetic
Comparison

Bibliography

1 Introduction

2 Assignment

3 Expression

- Arithmetic
- Comparison

4 Bibliography



Contents

Linear ACSE

Ettore
Speziale

Introduction

Assignment

Expression
Arithmetic
Comparison

Bibliography

1 Introduction

2 Assignment

3 Expression

- Arithmetic
- Comparison

4 Bibliography



Work-flow

Linear ACSE

Ettore
Speziale

Introduction

Assignment

Expression

Arithmetic Comparison

Bibliography

The LANCE files are a list of statements:

- `see Acse.y`

We have just seen a simple statement:

- the `write` statement

It is linear:

- no conditional
- translation depends only on `write` itself

Today we will see something close:

- assignments
- expressions



Interlude

Linear ACSE

Ettore
Speziale

Introduction

Assignment

Expression

Arithmetic

Comparison

Bibliography

Before going forward:

- how is it possible to generate instructions?

An helper function is associated to every instruction:

- it allows to emit the instruction hiding low level details
- see `axe_gencode.h`

Generating instructions

Instruction	Helper
ADD	<code>gen_add_instruction</code>
ADDI	<code>gen_addi_instruction</code>
READ	<code>gen_read_instruction</code>
BEQ	<code>gen_beq_instruction</code>



Contents

Linear ACSE

Ettore
Speziale

Introduction

Assignment

Expression
Arithmetic
Comparison

Bibliography

1 Introduction

2 Assignment

3 Expression

- Arithmetic
- Comparison

4 Bibliography



A Complete Statement

Linear ACSE

Ettore
Speziale

Introduction

Assignment

Expression

Arithmetic
Comparison

Bibliography

Consider the simple assignment $a = 4$:

- we want to copy 4 inside a
- we need both a (left-hand side) and 4 (right-hand side)

When do we know all the data needed?

- when the parser recognizes the `assign_statement` rule



Generalized Assignment I

Linear ACSE

Ettore
Speziale

Introduction

Assignment

Expression

Arithmetic
Comparison

Bibliography

Think at left-hand sides:

scalar stored in a register

array cell stored somewhere in the memory

Moreover:

- they have different syntax

And right-hand sides:

- just something evaluable to a scalar



Generalized Assignment II

Linear ACSE

Ettore
Speziale

Introduction

Assignment

Expression

Arithmetic
Comparison

Bibliography

Left-hand sides are too different:

- the rule must be specialized

Right-hand side are equal:

- should be factorized through the `exp` rule

Now, better to switch to code:

- look at the `assign_statement` rule in `Acse.y`
- scalars are stored into registers and manually handled ¹
- arrays are managed exploiting a function from `axe_array.h`

¹The `if` is explained later.



Contents

Linear ACSE

Ettore
Speziale

Introduction

Assignment

Expression

Arithmetic
Comparison

Bibliography

1 Introduction

2 Assignment

3 Expression

- Arithmetic
- Comparison

4 Bibliography



The Need to Type

Linear ACSE

Ettore
Speziale

Introduction

Assignment

Expression

Arithmetic
Comparison

Bibliography

Most of ACSE code deals with expressions:

- assignments
- arrays indexing
- conditionals

The exp has been typed to generalize expressions management:

Expression type ²

```
typedef struct t_axe_expression {  
    int value;  
    int expression_type;  
} t_axe_expression;
```

²See `axe_struct.h`.



Building Expressions

Linear ACSE

Ettore
Speziale

Introduction

Assignment

Expression

Arithmetic
Comparison

Bibliography

The expression framework:

- allows to combine expressions together
- generates code to compute expressions
- described in `axe_expressions.h`

They are built recursively:

- two base cases: IMMEDIATE and REGISTER expressions
- intermediate values kept into REGISTER expressions
- `create_expression` allows to build base expressions



Expression Values

Linear ACSE

Ettore
Speciale

Introduction

Assignment

Expression

Arithmetic

Comparison

Bibliography

The expression value is stored into the value field:

immediate the value of the immediate

register the register storing that expression

Un-boxing expressions

```
if($3.expression_type == IMMEDIATE)
    gen_addi_instruction(..., $3.value);
else
    gen_add_instruction(..., $3.value,
                        CG_DIRECT_ALL);
```



Add

Linear ACSE

Ettore
Speziale

Introduction

Assignment

Expression

Arithmetic

Comparison

Bibliography

Very simple expressions:

Adding two expressions

```
exp:
...
    | exp AND_OP exp {
        $$ = handle_bin_numeric_op(program,
                                     $1,
                                     $3,
                                     ANDB);
    }
...
```



Lesser Than

Linear ACSE

Ettore
Speziale

Introduction

Assignment

Expression

Arithmetic

Comparison

Bibliography

Relational operators handled with expressions too:

Comparing two expressions

```
exp:
...
  | exp LT exp {
    $$ = handle_binary_comparison(
          program, $1, $3, _LT_);
  }
...
```



Contents

Linear ACSE

Ettore
Speziale

Introduction

Assignment

Expression
Arithmetic
Comparison

Bibliography

1 Introduction

2 Assignment

3 Expression

- Arithmetic
- Comparison

4 Bibliography



Bibliography

Linear ACSE

Ettore
Speziale

Introduction

Assignment

Expression

Arithmetic
Comparison

Bibliography



A. Di Biagio and G. Agosta.

Advanced Compiler System for Education.

<http://compilergroup.elet.polimi.it>, 2008.



Formal Languages and Compilers Group.

Software Compilers.

<http://compilergroup.elet.polimi.it>, 2010.