# MicroTESK - Task #4832

## [translator][model] VLIW architectures support

04/04/2014 09:10 AM - Alexander Kamkin

| | | | |
|---|---|---|---|
| **Status:** | Closed | **Start date:** | 04/04/2014 |
| **Priority:** | Normal | **Due date:** | 05/31/2014 |
| **Assignee:** | Andrei Tatarnikov | **% Done:** | 100% |
| **Category:** | | **Estimated time:** | 0.00 hour |
| **Target version:** | 2.1 | **Spent time:** | 0.00 hour |
| **Detected in build:** | svn | **Published in build:** | 141018 |

**Description**

The key issue of VLIW architectures is composition of complex instructions (so-called *words*) from simple ones (so-called *syllables*). This can be done by enabling multiple operation parameters of the operation type:

```
op S1(x: arithmetic_instruction) ...
op S2(x: control_instruction) ...

op W2(x: S1, y: S2)
    syntax = format("{%s, %s}", x.syntax, y.syntax)
    image  = format("1101111010101101%b%b", x.image, y.image)
    action = { x.action; y.action; }

op W = W2 | ...

op instruction(x: W)
    syntax = x.syntax
    image  = x.image
    action = { x.action; }
```

Translation scheme seems to be simple:

```
public class W2 {
    private S1 x;
    private S2 y;
    ...
    public void action() {
        x.action();
        y.action();
    }
}

public class instruction {
    private W x;
    ...
    public void action() {
        x.action();
    }
}
```

Instantiation of complex instructions in test templates is done as follows:

```
    W2_begin
        S1 r(1), r(2), r(3)
        S2 r(4), r(5), r(6)
    W2_end
    ...
```

In general case it might look as it is shown below:

```
    Word_begin
        SubWord_begin
            Syllable_1 ...
```

```
            ...
        SubWord_end
        Syllable_N ...
    Word_end
    ...
```

If there is no ambiguity in determining how to compose words from syllables, begin/end brackets may be omitted:

```
    Syllable_1 ...
    ...
    Syllable_N ...
    ...
```

### History

#### #1 - 04/04/2014 09:37 AM - Alexander Kamkin

Better syntax:

```
    Word {
        SubWord {
            Syllable_1 ...
            ...
        }
        Syllable_N ...
    }
    ...
```

#### #2 - 04/04/2014 09:43 AM - Alexander Kamkin

So, there should be possibility of "manual" instantiation of instructions in test templates:

```
    instruction {
        ADD r(1), r(2), r(3)
    }
```

It does not make sense if all non-terminal instructions have one parameter per each. In this case, one can use the default form:

```
    ADD r(1), r(2), r(3) // The default builder Instruction_ADD is invoked
```

Metainformation should be extended.

#### #3 - 05/22/2014 05:03 PM - Andrei Tatarnikov

*- Status changed from New to Open*

#### #4 - 07/22/2014 10:28 AM - Andrei Tatarnikov

*- % Done changed from 0 to 40*

Сделана поддержка на уровне транслятора.

#### #5 - 09/02/2014 12:15 PM - Andrei Tatarnikov

*- % Done changed from 40 to 60*

#### #6 - 09/03/2014 11:24 AM - Andrei Tatarnikov

Синтаксис будет такой:

Сложный вызов (VLIW):

```
longword(
  (add r(1), r(2), r(3)),
  (sub r(1), r(2), r(3)),
  r(4),
  0xFF
)
```

Простой вызов (RISK):

```
add r(1), r(2), r(3)
```

Примерные сигнатуры runtime-методов, реализующих ISA:

```
def longword(a, b, c, d)
  puts "longword(#{a}, #{b}, #{c}, #{d})"
end

def r(i)
  "r#{i}"
end

def add(a, b, c)
  "add(#{a}, #{b}, #{c})"
end

def sub(a, b, c)
  "sub(#{a}, #{b}, #{c})"
end
```

#### #7 - 09/08/2014 05:57 PM - Andrei Tatarnikov

*- % Done changed from 60 to 80*

#### #8 - 09/10/2014 12:02 PM - Andrei Tatarnikov

*- % Done changed from 80 to 90*

Сейчас все существующие примеры работают. Остались мелкие доделки, тестирование, удаление кода старой реализации (все классы, работающие с сущностью Instruction).

#### #9 - 10/31/2014 04:44 PM - Andrei Tatarnikov

*- Status changed from Open to Closed*

*- % Done changed from 90 to 100*

*- Published in build set to 141018*

Сделано. Кусок примера для иллюстрации:

```
# Possible syntax styles to address the VLIW ISA:
# Style 1:

vliw(
  (addi r(4), r(0), 5  do situation('overflow') end),
  (addi r(5), r(0), 10 do situation('normal') end)
)

# Style 2:

vliw(
  addi(r(4), r(0), 5)  do situation('overflow') end,
  addi(r(5), r(0), 10) do situation('normal') end
)
```