CS4212 – Compiler Design
School of Computing
National University of Singapore

September 23, 2012

Problem Set 4

Semester 1, 2012/13

Due: ***October 14, 23:59*** ***Marks: 8***

Submission: In IVLE, in the cs4212 workbin, you will find a folder called "`Homework submissions`". In that folder, you will find ***3 new subfolders:*** **`PS3P01,…,PS3P03`**. The last two digits of the folder name indicate the solution that is to be submitted into that folder: the solution to *Question 1* into **`PS3P01`**, and so on (that is, you need to submit 3 separate solutions to 4 problems). A solution should consist of a ***single text file*** that can be loaded into the SWI-Prolog environment and executed. You should provide as much supplementary information about your solution as you can, ***in the form of Prolog comments***.

The authors' ***names and matriculation numbers should appear clearly in a comment*** at the top of the file. Teams may have at most 2 members. Each team should have a single submission for each problem; there is ***no need for multiple submissions of the same solution*** from each of the contributors. Please remember that for team submissions, each team member receives 75% of the marks awarded to the solution.

Problem Set 4 relies on a modification of the `comp_procedures.pro` compiler, named `comp_procedures_index.pro`, which you will find available from the IVLE Lesson Plan. This new compiler adds a new operator to our toy language. This operator allows a variable to be indexed, somewhat like an array, giving access to the variable's "neighbours". For example, consider the following program snippet:

```
global a,b,c ;
{ local x,y,z ;
      x = b@1 ; /* performs x = c */
      x = b @ -1 ; /* performs x = a */
      y = 3; z = 2 ;
      x = y @ (y-z) ; /* performs x = z */
}
```

In general, the index expression `var@expr` is evaluated in the following way. The expression `expr` is first evaluated, and that value is multiplied by 4, and then added to the address of `var`. The resulting address is dereferenced, obtaining the value of the index expression. This kind of expression can appear embedded in a larger expression, but only

allows reading access. With this facility, we can have a rudimentary implementation of read-only arrays, as showcased by the following toy program:

```
/* Compute the minima of two improvised arrays */
global i,a,b,c,d,e,f,min1,min2 ; /* a,...,f form a 6-elem array */
fmin#(void) ::
{ local i,x,x1,x2,x3,x4,x5 ; /* x,x1,...,x5 form a 6-elem array */
  x = 9 ; x1 = 7 ; x2 = 4 ; x3 = 6 ; x4 = 8 ; x5 = 10 ;
  i = 0 ; min1 = 100 ;
  while i < 6 do
  { if min1 > x@i then { min1 = x@i } ;
    i = i + 1
  }
} ;
fmin#(void);
a = 90 ; b = 59 ; c = 30 ; d = 45 ; e = 23 ; f = 94 ;
min2 = 100 ; i = 0 ;
while i < 6 do
{ if min2 > a@i then { min2 = a@i } ;
  i = i + 1 ;
}
```

Problem 1 [C][4 marks, submit to `PS3P01`]

Add a "real" implementation of arrays to the toy language. The elements of your implementation should be the following:

- Global array declaration: `{ global ...,a@size,... ; ... }` (where `size` is a constant)
- Local array declaration: `{ local ...,a@size,... ; ... }` (where `size` is a constant)
- Array assignment: `a@i = expr ;`
- Global arrays must be printed at the end of the program

Hint: You only need to make the following changes to `comp_procedures_index.pro`:

- Definition of `local_vars` predicate.
- Definition of `global_vars` predicate.
- Add a rule to the `cs` predicate with the head `cs((A@I=E),Code,Ain,Aout):-...`
- Change the runtime-stmt.c so that it prints arrays.
- In the Pentium assembly language, the directive `.space N` reserves `N` bytes of data, and can be used to allocate an array (`N` must be `size*4`).
- Use the `leal` assembly language instruction to compute the address of an array element.

The following sorting program should compile and run correctly:

```
global a@10,i ;
a@0=10 ; a@1=5 ; a@2=2 ; a@3=7 ; a@4=9 ; a@5=3 ; a@6=7 ; a@7=6 ; a@8=8 ;
a@9=4 ;
i = 0 ;
while i < 10 do {
    local j ;
    j = i+1 ;
    while j< n do {
        if a@i > a@j then {
            local tmp ;
            tmp = a@i ; a@i = a@j ; a@j = tmp ;
        } ;
        j = j + 1 ;
    }
    i = i + 1 ;
}
```

You should also try a similar program where the array is local. Now, since there are no array bounds checks, you should pay close attention to the array accesses, since you can easily clobber memory content that is outside the user data area, and which may bring your program into an unstable state.

Problem 2 **[C][2 mark, submit to PS3P02]**

On top of your solution to Problem 1, implement vectorial initialization, in the form:

`a = (expr1,expr2,...,exprk)`

In this statement, assume that `a` is an array of size `k`, and simply initialize `k` 4-byte cells starting at the address of `a` with the elements in the tuple, after they have been evaluated. You do not need to check that `a` has indeed size `k`. The expressions in the tuple may refer back to elements of `a`. For instance,

`a = (a@1,a@0)`

swaps the first two elements of array `a`. Hint: you only need to add a new rule to the `cs` predicate, with the heading `cs((A=(X,Y),Code,Ain,Aout) :-...`

Problem 3 **[C][2 marks, submit to PS3P03]**

On top of your solution to Problem 2, implement array segment assignment in our toy language. The general format of this statement is:

`a@(Expr1:Expr2) = b@(Expr3:Expr4)`

The expression `a@(Expr1:Expr2)` denotes the array segment of `a` that starts at an index equal to the value of `Expr1` (inclusive), and ends at an index whose value is `Expr2` (exclusive). A similar meaning is attached to `b@(Expr3:Expr4)`. The assignment is correct semantically only if both segments have the same length. The effect of this assignment is identical to the following program snippet:

```
{ local tmpidx1, tmpidx3, iter, limit ;
  tmpidx1 = Expr1 ; tmpidx3 = Expr3 ;
  iter = 0 ; limit = Expr2 – Expr1 ;
  while iter < limit do
     { a@(tmpidx1+iter) = b@(tmpidx3+iter) ; iter = iter + 1 }
}
```

Hint: you only need to add a new rule to the `cs` predicate, with the heading
`cs((A@(E1:E2)=B@(E3:E4)),Code,Ain,Aout) :-...`