Conclusions of the 14^{th} International Real-Time Ada Workshop

Stephen Michell, *Session Chair*
Maurya Software Inc, Ontario, Canada
stephen.michell@maurya.on.ca

Jorge Real, *Rapporteur*
Universidad Politécnica de Valencia, Spain
jorge@disca.upv.es

1 Introduction

The last session of IRTAW-14 was devoted to concluding on the results of the workshop, with the goal of prioritizing and selecting Ada Issues (AIs) to be produced and sent to the ISO/IEC JTC1/SC22/WG9 Ada Rapporteur Group (ARG). It allowed time for closing some open issues.

In this report, Sections 2 to 4 summarize the final discussion around some open issues. Section 5 reflects the list of AIs to be produced by the workshop. The plan for next meeting is considered in Section 6. There is a final consideration about concurrency vulnerabilities in Section 7. Finally, Section 8 concludes with the closing of IRTAW-14.

2 Barrier suspension objects

Stephen Michell summarized the proposal for Ada to include barrier suspension objects to allow parallel release of multiple readers upon a certain condition, expressed by means of a barrier [4]. This proposal is targeted to multiprocessor architectures and the goal is to allow true parallelism when multiple readers wait for data produced by a single writer task. In such cases, the maximum efficiency is achieved by broadcasting the data in parallel to all the interested reader tasks. This behavior cannot be accomplished by means of an entry, as presently specified in Ada, due to the fact that only the first waiting task would be released upon barrier opening and then the barrier condition would need to be reevaluated every time a single task is served. The proposal can be supported in POSIX and in most present hardware implementations of a barrier.

It was noted that the definition of barrier suspension objects should be accompanied by pragma Preelaborate. The workshop did not see any need for requiring barrier suspension objects to be declared at library level. No implications were identified with respect to pragmas Intrinsic and Inline. There was unanimous support for the proposal.

3 Named storage pools

Named storage pools were proposed in [5]. They are motivated by the convenience to use storage pools specifically tied to one of the different kinds of memory available, since memory maps include different memory technologies in many systems (RAM, ROM, FLASH, etc.).

The workshop however did not find enough motivation for pushing for this change to the language: the proposal is not mature enough and there were opinions in the sense that there may be ways to achieve a similar functionality in current Ada. The proposal was therefore withdrawn. It was agreed, however, that there should be syntax added to Ada to permit the specification of an address for a declared storage pool.

4 Execution time control for interrupt handling

Kristoffer Gregertsen gave a summary of the proposal to introduce mechanisms for monitoring the execution time spent in servicing interrupts. This proposal was based on [3] and [1]. The proposed API (i) defines one execution-time clock per `Interrupt_ID`, (ii) allows the mechanism to obtain the time spent in the handling of each interrupt, and (iii) also allows association of timers to those clocks.

Use of `Ada.Interrupts.Interrupt_ID` was preferred to using `Task_ID` to identify execution time of the different interrupt handlers. A new package `Ada.Execution_Time.Interrupts` contains the following subprogram:

```
function Clock (I: Ada.Interrupts.Interrupt_ID) return CPU_Time;
```

The function returns the execution time spent in handling the identified interrupt, or returns `CPU_Time_First` if the facility is not supported by the implementation.

The proposal was supported by 17 votes for, no vote against, and 3 abstentions. Hence an AI will be produced on this topic. Note that being a child package of `Ada.Execution_Time`, its implementation would be optional.

The workshop then considered a natural extension to this facility: timers for CPU time spent in interrupt handling. This feature would need the inclusion of interrupt clocks first, since timers rely on clocks. Although there was no objection to the interface described in [3], there was no general support for pushing this feature forward to standardization (4 votes for, 2 against and 13 abstentions). The workshop however agreed to suggest to ARG the inclusion of an implementation advice stating that this service, if implemented, should stick to the proposed interface:

```
with Ada.Interrupts;
package Ada.Execution_Time.Timers.Interrupts is
   type Timer (I: Ada.Interrupts.Interrupt_ID)
      is new Ada.Execution_Time.Timers.Timer(
         Ada.Task_Identification.Null_Task_Id'Access)
      with private;
private
   . . .
end Ada.Execution_Time.Timers.Interrupts;
```

5 Wrapping up

Alan Burns prepared the list of topics about which the workshop agreed to produce new AIs. The list was reviewed and the different items were assigned to those in charge of writing them. The final list considers:

1. Addition of affinity support packages, interrupt affinities and considerations about spin locking — A. Burns and A. Wellings.
2. Change definition of group budgets to include processor, with default to processor 1 — A. Burns and A. Wellings.
3. Addition of an implementation advice to allow for multiprocessor execution of Ravenscar programs — J. Ruiz.
4. Addition of timers to the Ravenscar profile (a maximum of one timer per task) — T. Vardanega. [1]
5. Add the definition of barrier suspension objects — S. Michell.
6. Implementation advice on interrupt monitoring — M. González and M. Aldea.
7. Addition of operations *yield* and *yield to higher priority* in non-preemptive scheduling — A. Burns.
8. Deadlines in synchronous task control — A. Burns.
9. Addition of interrupt execution-time accounting clocks — M. González.

6 Conclusion and next IRTAW

Deadlines were set for finalization of session reports, production of final versions of the position papers, and writing of the AIs to be sent to ARG. Alan Burns will centralize the AIs and propose them in the next ARG meeting.

There was general agreement about the need of future editions of IRTAW. The next edition will be organized by Michael González in the Santander area, in Spain. The workshop is scheduled for April or May 2011, hence we leave some 18 months between editions 14 and 15. Mario Aldea will head the role of Program Committee Chair.

[1] Tullio will check that the Ravenscar model is not broken with this addition and get feedback from implementors — perhaps add a restriction (e.g. `Max_Nr_Of_Timers_Per_Task` and set it to 1 for Ravenscar.

7 Consideration of concurrency vulnerabilities

The ISO/IEC JTC 1/SC 22/WG 23 (WG23, for short) is preparing a technical report about Programming Language Vulnerabilities. One of the items in the workshop agenda was to note the absence of concurrency-related vulnerabilities in the technical report being prepared, as reflected in the position paper [2].

The workshop decided to submit this position paper to WG23, after receiving comments and suggestions for improvement from participants at the workshop. Miguel Pinho noted that multiprocessor execution may be yet another source of vulnerabilities worth considering.

8 Closing

There being no other pending issues, Stephen Michell closed the session and the workshop. The workshop thanked specially the presence of first-time participants and encouraged them to continue to do so. All thanked Tullio Vardanega for the splendid local arrangement.

References

[1] M. Aldea and Michael González-Harbour. Execution time monitoring and interrupt handlers. *Ada Letters*, This issue.

[2] A. Burns and A. Wellings. Language vulnerabilities - let's not forget concurrency. *Ada Letters*, This issue.

[3] K. Gregertsen and A. Skavhaug. Execution-time control for interrupt handling. *Ada Letters*, This issue.

[4] S. Michell, L. Wong, and B. Moore. Real-Time paradigms needed post Ada 2005. *Ada Letters*, This issue.

[5] L. Wong, S. Michell, and B. Moore. Named memory pool for Ada. *Ada Letters*, This issue.