Annex - Formalisation

Our report proposes several hypotheses and corresponding matching solutions we developed. This document provides a formalised version of the solutions detailed in the report.

Definitions

1. Let P be a **program** executed on RISC-V. Its execution corresponds to a **sequence of instructions** executed by the processor. We make the assumption that an instruction is uniquely identified by a date d. The sequence of dates, corresponding to the sequence of instructions, is called a **trace** t.
2. Let $d_1 < d_2$ means the date d_1 happens before the date d_2, i.e., $<$ is a total order.
3. Let M be the **space of memory addresses** in the trace t.
4. Let $\perp$ be the **null address** (i.e. no address).
5. Let $addr(d) \in M \cup \{\perp\}$ be the **memory address** impacted by instruction at date d.
   (a) Let $read(d) \in M \cup \{\perp\}$ be the **memory address accessed for reading**.
   Thus, $\forall d$, $read(d) \neq \perp \Leftrightarrow d$ is a load instruction.
   (b) Let $write(d) \in M \cup \{\perp\}$ be the **memory address accessed for writing**.
   Thus, $\forall d$, $write(d) \neq \perp \Leftrightarrow d$ is a store instruction.
   (c) Let $jump(d) \in M \cup \{\perp\}$ be the **address jumped to**.
   Thus, $\forall d$, $jump(d) \neq \perp \Leftrightarrow d$ is a jump instruction.
6. Let $rs(d) \in [\![0; 31]\!] \cup \{\perp\}$ be the **source register** used at time d.
7. Let $rd(d) \in [\![0; 31]\!] \cup \{\perp\}$ be the **destination register** used at time d.
8. Let $wmethod(d) \in \{sp, other\}$ be the **method** used by instruction d to write in memory.

Predicates

1. Let $read_b(d)$ be a boolean so that $read_b(d) = true \Leftrightarrow read(d) \neq \perp$.
2. Let $write_b(d)$ be a boolean so that $write_b(d) = true \Leftrightarrow write(d) \neq \perp$.
3. As a consequence, we have $addr(b) \neq \perp \Leftrightarrow read_b(d) \vee write_b(d)$.
4. Let $reads(d_1, d_2)$ be the **set of dates where load instructions are executed**.
   Thus, $reads(d_1, d_2) = \{d \mid d \in [d_1; d_2] \wedge read_b(d)\}$.
5. Let $writes(d_1, d_2)$ be the **set of dates where store instructions are executed**.
   Thus, $writes(d_1, d_2) = \{d \mid d \in [d_1; d_2] \wedge write_b(d)\}$.
6. Let $dates_T(R) = [min(R); max(R) + T]$.
7. Let $addrs(R) = \{addr(d) \mid d \in R\}$.
8. Let $achunk(R) = [min(addrs(R)); max(addrs(R))]$.
9. Let $int_{T,N}(R)$ be a boolean for **interval of consecutive addresses** with parameters N the **minimum size of the set** and T the **maximum timespan between two dates** so that:

$$int_{T,N}(R) = \left| \begin{array}{ll} |R| > N & \wedge \\ \forall d \in R, \quad (\forall d' \in R,\ d \leq d') \vee (\exists d' \in R,\ d - d' < T) & \wedge \\ \forall d \in R, \quad addr(d) \neq \perp & \wedge \\ \forall a \in M, a \in achunk(R) \Leftrightarrow a \in addrs(R) & \wedge \\ \forall d_1, d_2 \in R, \quad d_1 < d_2 \implies addr(d1) < addr(d2) & \end{array} \right.$$

Formalisation per hypothesis

Here, we focus on situations where the hypotheses from the report are not satisfied. We formalise the conditions needed for a flag to be raised.

Hypothesis 2. Program data and control flow data should not be mixed. T, N, D are parameters with $(D > T)$.

As illustrated on Figure 1, a flag is raised if $\exists\, a, b, R, d_r, d_j$,

$$
\begin{array}{|ll}
writes(a, b) = R & \wedge \\
int_{T,N}(R) & \wedge \\
max(R) + T < d_r < d_j \leq max(R) + D & \wedge \\
read(d_r) \in addrs(R) & \wedge \\
jump_b(d_j) & \wedge \\
rs(d_j) = rd(d_r) &
\end{array}
$$

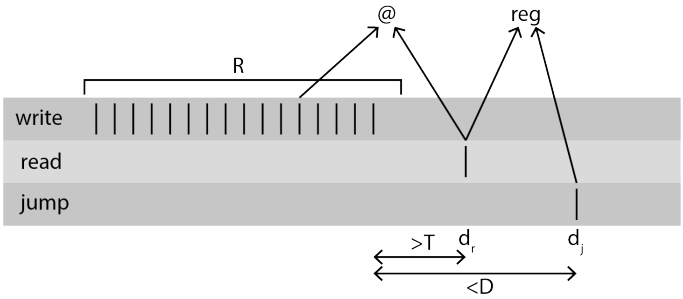

Figure 1: Illustration of a trace where a flag is raised.

Hypothesis 3. Each pointer should be processed separately from each other. T, N, D are parameters with $(D > T)$.

As illustrated on Figure 2, a flag is raised if $\exists\, a, b, d_1, d_2$

$$
\begin{array}{|ll}
writes(a, b) = R & \wedge \\
int_{T,N}(R) & \wedge \\
max(R) + T < d_1 < d_2 \leq max(R) + D & \wedge \\
read(d_1) \in addrs(R) & \wedge \\
read_b(d_2) & \wedge \\
rs(d_2) = rd(d_1) &
\end{array}
$$

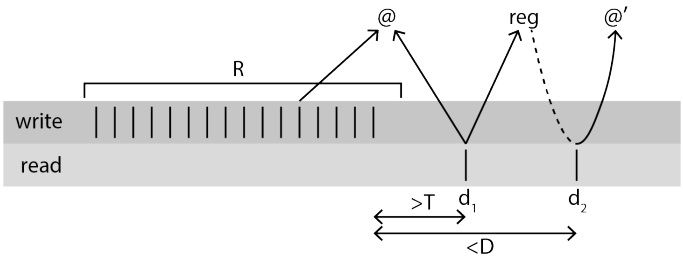

Figure 2: Illustration of a trace where a flag is raised.

Hypothesis 4. The processing of a buffer cannot extend past the boundaries of a buffer. T, N, D are parameters with $(D > T)$.

As illustrated on Figure 3, a flag is raised if $\exists\, a_1, b_1, a_2, b_2, R_1, R_2, d$,

$$\begin{array}{|ll} writes(a_1, b_1) = R_1 \wedge int_{T,N}(R_1) & \wedge \\ reads(a_2, b_2) = R_2 \wedge int_{T,N}(R_2) & \wedge \\ d \in R_2 & \wedge \\ max(dates(R_1)) + T < min(dates(R_2)) < d < max(dates(R_1) + T + D & \wedge \\ min(addrs(R_1)) \neq min(addrs(R_2)) & \wedge \\ read(d) = min(addrs(R_1)) & \end{array}$$

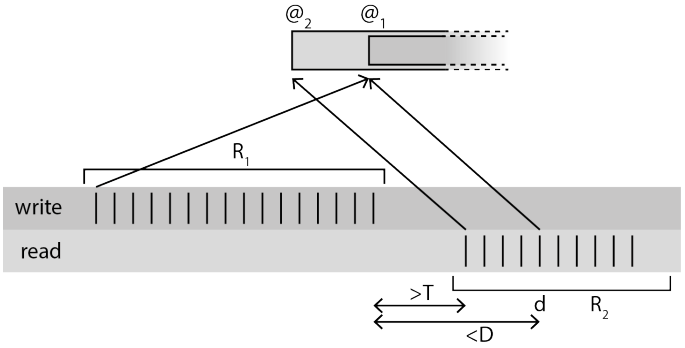

Figure 3: Illustration of a trace where a flag is raised.

Discarded Hypothesis 5. The compiler, through the symbol table, will always use the same strategy to access a local variable. D is a parameter.

A flag is raised if $\exists\, d_1, d_2$,

$$\begin{array}{|ll} write(d_1) = write(d_2) & \wedge \\ wmethod(d_1) \neq wmethod(d_2) & \wedge \\ d_2 - d_1 < D & \end{array}$$