CS250P Computer Systems Architecture

Homework 1, Due: November 8th, 222

Submit via gradescope! Link will be available soon.

Question 1: Register Windows Analysis

"Register Window" [1] is a computer architecture concept which assigns a subset of architecturally available registers to each subroutine/function. It is not used in x86, RISC-V, or ARM. But it was a prominent feature in SUN SPARC and Intel Itanium. SPARC is a popular ISA in the European space program for radiation-hardened processors [2].

In a register window implementation, the actual hardware includes more hardware registers than what is visible to the software at any given time. For example, a 32-bit SPARC ISA defines three sets of 8 registers visible to the software: IN, LOCAL, and OUT, as seen in Figure 1. Assembly code can only address these 24 registers, using notation such as IN[4], and LOCAL[0], and also the resulting binary only encodes these 24 registers.

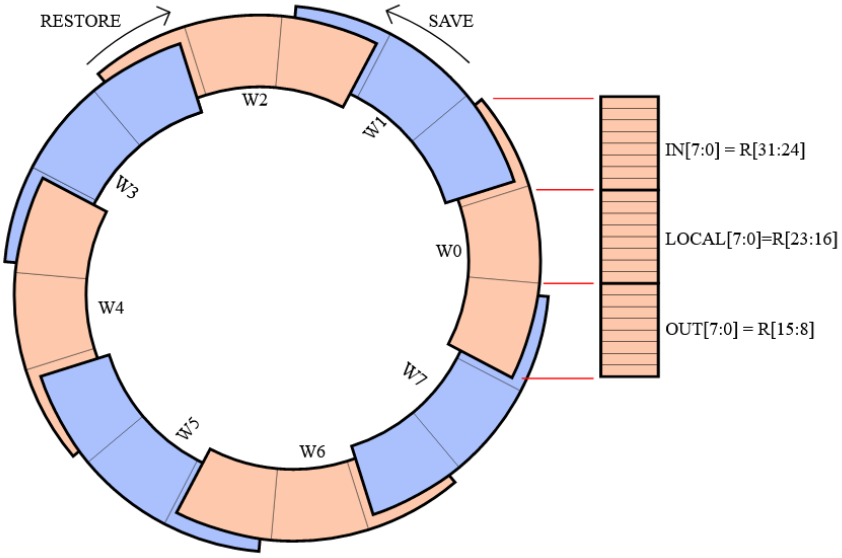

Figure 1: Register windows implementation in 32-bit SPARC.

But the hardware actually includes more than 24 registers. In fact, the 32-bit SPARC processor implemented eight 8-register sets, resulting in 64 registers in total, organized in a ring, as seen in the left of Figure 1. Whenever a subroutine calls another subroutine, the "window" is moved forward, so that the "OUT" of the caller function can be accessed as "IN" of the callee function. From the function software's point of view, the function call parameters are always available in the IN registers, and when it must call functions of its own, the function call parameters should be stored in the OUT registers.

This is a very different approach from what we are used to from x86, RISC-V, ARM, or MIPS, all of which shares the same set of registers between subroutines. This approach has its pros and cons, which we will analyze.

Question 1.1: ISA Scalability: Given two implementations of SPARC, one with only four sets of registers in the circular register file, and one giant one with 32 sets, can you run the same compiled binary on both processors? (Yes/No)

Question 1.2: Why or why not?

Question 1.3: Given two processors with and without register windows but with the same number of total physical register in silicon, which would typically require a smaller number of bits to encode the register index in an assembly instruction? (Windows/No windows)

Question 1.4: Why or why not?

Question 1.5: Given a RISC-V processor with 32 registers, and the 8-set SPARC processor shown in Figure 1, which do you think will likely support a faster clock speed? (RISC-V/SPARC)

Question 1.6: Why?

Question 1.7: Given a RISC-V processor with 32 registers, and a SPARC processor with the same number of registers (32 registers, four sets of 8), which do you think will use each register more efficiently? Think about what happens with small recursive function calls. (RISC-V, SPARC)

Question 1.8: Why?

Question 2: Pipelining

Consider a classical 5-stage RISC pipeline, but with the order of Memory and Execute stages reversed, as seen in Figure 2.

Figure 2. Classical RISC pipeline with some shuffled stages.

Question 2.1: In the case of a branch mis-predict, will this design complicate the handling of branch mis-predicts? (Y/N)

Question 2.2: Why or why not?

Question 2.3: Assuming a program with no branch instructions but with a lot of memory access instructions, will this new pipeline harm performance? (Y/N)

Question 2.4: Why or why not?

References

[1] Understanding stacks and registers in the Sparc architecture(s), https://cseweb.ucsd.edu/~gbournou/CSE131/sparcstack.html

[2] ESA - Microprocessors https://www.esa.int/Enabling_Support/Space_Engineering_Technology/Onboard_Computers_and_Data_Handling/Microprocessors