Register window ARC

Dennis Schroer (s1228838)
Nick ten Veen (s1223631)

Inleiding

Het uitvoeren van functies is bij de ARC processor vrij inefficiënt. Als er een functie wordt uitgevoerd dan moeten eerst de variabelen en het return adres op de stack geplaatst worden om het mogelijk te maken om terug te kunnen keren naar waar het programma gebleven was. Om dit proces te stroomlijnen en te vereenvoudigen is het wenselijk om een register window te maken. Hierbij is maar een gedeelte van de registers zichtbaar waardoor de gebruiker een soort van scoping heeft bij elke call. Elke keer als de gebruiker een call maakt schuift de register window op naar rechts waardoor de gebruiker weer andere registers ziet om in te werken. Het achterste stuk van de window wordt het eerste stuk van de nieuwe window als deze opschuift, waardoor er variabelen meegegeven kunnen worden aan een functie.

Om de register window te implementeren is het noodzakelijk om de ARC processor aan te passen. Er zullen wat aanpassingen in het microprogramma gemaakt moeten worden waardoor de huidige register window correct wordt bijgehouden.

Aanpak

Om bij te kunnen houden waar de huidige register window zit, introduceren we een register waarin de huidige window offset wordt bijgehouden: de current window pointer (CWP). Dit register zal in de implementatie in register %r38 zitten.

De eerste 8 registers zijn voor de gebruiker statische registers die in elke window zichtbaar zijn. De ARC processor gebruikt de registers %r32 tot en met %r38 voor de program counter, instructie register en wat tijdelijke registers voor het microprogramma. Om de window wat gemakkelijker te implementeren worden deze registers naar het begin verplaatst, direct achter de eerste 8 statische registers. Deze verplaatsing is niet zichtbaar voor het microprogramma. Pas op het laatste moment worden de registeradressen omgezet naar hun daadwerkelijke positie. Door deze aanpassing kan de window lineair verplaatsen over de registers %r16 tot en met register $n \cdot 16 + 17$, waarbij n gelijk is aan de window diepte.

Om te kunnen bepalen wat de huidige window is, zal register %r38 gebruikt worden om de CWP bij te houden. De waarde van dit register wordt gebruikt om de offset van de huidige window te bepalen. Deze bepaling wordt net als de andere registers op het laatste moment gedaan. Hierdoor is het voor de gebruiker niet zichtbaar wat de huidige window op dit moment is. Ook het microprogramma zal maar 39 registers kunnen zien. De hardware zorgt ervoor dat de juiste registers aan de hand van de CWP aangesproken zullen worden.

Doordat de gebruiker en het microprogramma niet weten wat de huidige window is, kan het voor komen dat er een overflow of een underflow van de huidige window plaatsvindt zonder dat de gebruiker daarvan op de hoogte is. Om hier een voorziening in te kunnen regelen zijn er nog twee status bits: window_ov en window_un. In de huidige implementatie kan de gebruiker deze bits niet uitlezen, maar deze zou eventueel nog in een latere uitbreiding toegevoegd kunnen worden.

Het microprogramma moet op twee plekken aangepast worden: bij de CALL instructie en de JMPL instructie. Elke keer als er een call instructie is moet de CWP met 1 opgehoogd worden. Deze ophoging kan met een enkele instructie gedaan worden doordat de ALU een increment instructie heeft.

De JMPL instructie heeft nog een extra instructie nodig. Na een jmpl moet de CWP verlaagd worden met 1. De ALU heeft geen decrement instructie, dus zal hij moeten optellen met -1. Om -1 te krijgen ORN'en we %r0 met %r0 waardoor je 0xFFFFFFFF krijgt, wat in 2's complement -1 is.

Om de implementatie te testen zijn er drie kleine programma's gemaakt

- window_normal.asm --normaal programma zonder overflows
- window_overflow.asm --programma waarbij CWP overflowt
- window_underflow.asm --programma waarbij CWP underflowt