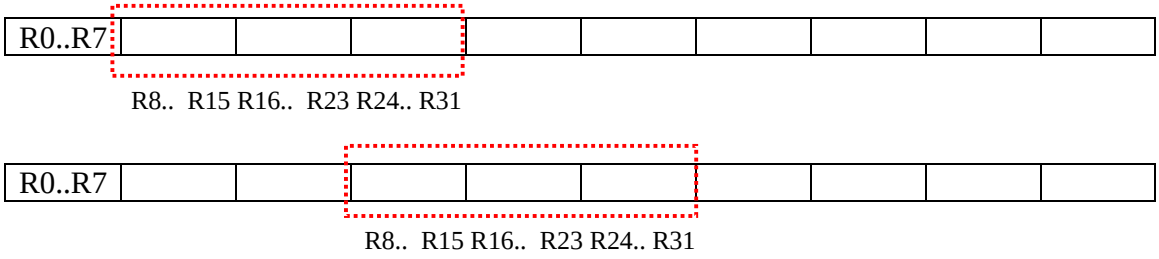
Implementatie van Register Windows

Bij een subroutineaanroep kunnen parameters op verschillende manieren worden doorgegeven (paragraaf 4.7). Dit doorgeven kan bijvoorbeeld door de data eerst op de stack te plaatsen in het geheugen en vervolgens wordt door de aangeroepen subroutine deze data weer opgehaald. In plaats van het slepen met data is voor de SPARC processor een trucje toegepast (zie paragraaf 6.8).

Het idee is verrassend eenvoudig. De gebruiker ziet maar 32 zichtbare registers maar er zijn er veel meer!
De locatie van de registers R8 t/m R31 is variabel (de gebruiker ziet dit niet). In onderstaand schema is dit weergegeven.



(zie ook figuur 6-19, pag. 236)
Bij een functieaanroep moet er voor worden gezorgd dat de data die je mee wilt geven aan de aan te roepen functie in de registers R24..R31 komt te staan.
Na de aanroep verschuift venster naar “rechts” over het fysiek aanwezig geheugen en zit de meegegeven data in R8 t/m R15.
Bij een programma is het gebruikelijk dat bij een CALL het terugkeeradres in %r31 gezet en is dan in de aangeroepen functie beschikbaar in %r15.
(Voor de registers R31 t/m R37 geldt hetzelfde als R0 t/m R7; deze registers schuiven niet mee; let er dus wel op dat ze niet direct aansluitend in het geheugen worden geplaatst).

Er zijn (tenminste) de volgende registers nodig:

- CWP, Current Window Pointer. Hiermee wordt het window bijgehouden. Bij een call/jmpl instructie wordt deze respectievelijk opgehoogd of afgelaagd.
- Verder zijn er nog twee bits; window_ov (window_overflow). D.w.z. dat alle registers in gebruik zijn. Bij een nieuwe CALL moet dan uiteaard weer de stack worden gebruikt. Uiteraard is er ook een window_un (window_underflow) bit.

Normaal is CWP een onderdeel van de PSR. Maar in mijn implementatie is de PSR slechts 4 bits breed (status). Wellicht is dan ook handiger om registers aan het ontwerp toe te voegen i.p.v. het uitbreiden van de PSR.
Er al window_ov optreedt zou eigenlijk de data van de minst recent gebruikte window op de stack moeten worden gezet (trap). Dit doen wij niet!! In deze opdracht laten we de boel dan in het honderd lopen omwille van de eenvoud. (Dus bij het testen van je ontwerp hier even op letten dat de recursiediepte niet meer is dan wordt ondersteund door de register window).

Opdracht

Maak de nodige aanpassingen in het ARC processor voor een register window diepte van 8 (of nog mooier generiek!). Maak ook een daarbij behorend testprogramma waarbij duidelijk een aantal call met returns en een windows_ov/windows_un optreedt. Er hoeft geen trap routine te worden ontwikkeld indien windows_ov/windows_uv wordt gezet.