Formalizing-ASM-Instructions

Bhagyashree Rane, Sai Ghule, Shravasti Deore, Uday Khedkar

December 2021

$$\textit{[Load Integer Immediate]}\frac{S(li, o1, o2) : \tau^S_{I,mv} \qquad o1 : \tau^S_{O,reg} \qquad o2 : \tau^S_{O,int}}{H; D; C; R \vdash S(li, o1, o2) \rightarrow H; D; C; R[o1 \mapsto o2]} \tag{1}$$

- If the register is a gpr or an fpr is not specified.
- There was no need to specify that o1, o2...so on are operands (O) while specifying types of the operands (eg. $o1 : \tau_{O,reg}$) beacause register is an operand(O) and not an instruction(I).

$$\textit{[Load Word]}\frac{S(lw, o1, o2) : \tau^S_{I,MV:lw} \qquad o1 : \tau^S_{O,R:gp} \qquad o2 : \tau^S_{O,D:addr}}{H; D; C; R : gp \vdash S(lw, o1, o2) \rightarrow H; D; C; R : gp[o1 \mapsto D[(o2, o2 + 4)]]} \tag{2}$$

- Each instruction(I)has been categorized into MV, CMP, CF, and so on. Each of the instruction types MV, CMP, CF have instructions like MV has lw, sw, li.d etc.
- Only R has the explicit need to be specified as a general purpose register, floating point register etc. $H; D; C; R : gp$ instead of the $H; D; C; R$ tuple complicates @ndbfms.

$$\textit{[Load Word]}\frac{S(lw, o1, o2) : \tau^S_{I,mv} \qquad o1 : \tau^S_{O,R_{gp}} \qquad o2 : \tau^S_{O,addr}}{H; D; C; R_gp \vdash S(lw, o1, o2) \rightarrow H; D; C; R_{gp}[o1 \mapsto D[(o2, o2 + 4)]]} \tag{3}$$

- Subscript of subscript has been introduced.
- It Complicates the $H; D; C; R$ tuple making it $H; D; C; R_{gp}$ or $H; D; C; R_{fp}$

$$\textit{[Load Word]}\frac{S(lw, o1, o2) : \tau^S_{I,MV:lw} \qquad o1 : \tau^S_{O,R:gpr} \qquad o2 : \tau^S_{O,D:addr}}{H; D; C; R \vdash S(lw, o1, o2) \rightarrow H; D; C; R[o1 \mapsto D[(o2, o2 + 4)]]} \tag{4}$$

- If the register is a gpr or an fpr is not specified.
- There was no need to specify o1, o2...so on, are operands (O) while specifying types of the operands (eg. $o1 : \tau_{O,reg}$) as register is an operand(O) and not an instruction(I) etc.

$$\textit{[Load Float Double]}\frac{S(l.d, o1, o2) : \tau^S_{mv} \qquad o1 : \tau^S_{fpr} \qquad o2 : \tau^S_{addr}}{H; D; C; R.d \vdash S(l.d, o1, o2) \rightarrow H; D; C; R.d[o1 \mapsto D.d(o2)]} \tag{5}$$

$$\textit{[Load Float Double]}\frac{S(l.d, o1, o2) : \tau^S_{I,mv} \qquad o1 : \tau^S_{O,R_{fp}} \qquad o2 : \tau^S_{O,addr}}{H; D_d; C; R_{fp} \vdash S(l.d, o1, o2) \rightarrow H; D_d; C; R_{fp}[o1 \mapsto D_d(o2)]} \tag{6}$$

- $D.d$ and D_f to get rid of need to specify the low address and high address of the data section D
- $D.d$ and D_f specifies exactly what the data type of the operand is.
- eg. $D.d$ and D_f specifies that the operand is of type double and that the low address $addr$ and high address $addr + 8$ are not mentioned explicitly
- $D.w$ and D_i specifies that the operand is of type word, therefore, the lower address is $addr$ and higher address is $addr + 4$
- But $D.w$ and D_i can be misinterpreted as the data segment D is different for different data types (eg. $D.w, D.d$) which isnt the case as the data segment.