Exemple de règles simplifiées de génération de code pour les fonctions.

Code CISC Load/Store

main(void) {	main_ LDW SP, #STACK_ADRS // charge SP avec STACK_ADRS LDQ NIL, BP // charge BP avec NIL=0 STW BP, -(SP) // empile le contenu du registre BP LDW BP, SP // charge contenu SP ds BP

*funct***(int x, int y) {**	*funct*_ STW BP, -(SP) // empile le contenu du registre BP
	LDW BP, SP // charge contenu SP ds BP
	
	// calcule aussi le dép. de chaque paramètre x_disp et y_disp

int a **;** **/*variable*/**	ADQ -INT_SIZE, SP // SP - INT_SIZE -> SP // aussi: calculer et garder le déplacement a_disp // (dit aussi offset) de la variable a dans le compilateur
z **=** <expression> **;**	<code du calcul de l'expression, résultat dans R0> STW R0, (BP)z_disp

appel de fonction	empilage de chaque paramètre x:
	<code du calcul de son expression, résultat dans R0>
	STW R0, -(SP)
	// aussi: calculer et garder le déplacement x_disp (dit aussi
	// offset) du paramètre x dans le compilateur
	// et calculer la taille totale param_size des paramètres
	JSR @funct // appel de la fonction fun d'adresse funct
	
	ADI SP, SP, #param_size // nettoyage paramètres

return <expression>	<code de calcul de l'expression, résultat dans R0> LDW SP, BP // abandon infos locales LDW BP, (SP)+ // charge BP avec ancien BP RTS // retour au programme appelant
} **/*accolade fermante finale de main */**	LDW SP, BP // abandon infos locales LDW BP, (SP)+ // charge BP avec ancien BP TRP #EXIT_EXC // EXIT: arrête le programme JEA @main_ // saute à main_ si redemande exécution
} **/*accolade fermante finale de fonction */**	LDW SP, BP // abandon infos locales LDW BP, (SP)+ // charge BP avec ancien BP RTS // retour au programme appelant

Code RISC strict Load/Store

main(void) {	main_ LDW SP, #STACK_ADRS // charge SP avec STACK_ADRS LDQ NIL, BP // charge BP avec NIL=0 ADQ -2, SP // décremente SP STW BP, (SP) // empile le contenu du registre BP LDW BP, SP // charge contenu SP ds BP

*funct***(int x, int y) {**	*funct_* ADQ -2, SP // décremente SP
	STW BP, (SP) // empile le contenu du registre BP
	LDW BP, SP // charge contenu SP ds BP
	
	// calcule aussi le dép. de chaque paramètre x_disp et y_disp

int a **;** **/*variable*/**	ADQ -INT_SIZE, SP // SP - INT_SIZE -> SP // aussi: calculer et garder le déplacement a_disp // (dit aussi offset) de la variable a dans le compilateur

z **=** <expression> **;**	<code du calcul de l'expression, résultat dans R0>
	// **STW R0, (BP)z_disp**
	LDW WR, #z_disp
	ADD BP, WR, WR
	STW R0, (WR)

appel de fonction	// empilage de chaque paramètre x:
	<code du calcul de son expression, résultat dans R0>
	ADQ -2, SP // décrémente SP
	STW R0, (SP) // sauvegarde contenu de R0 sur la pile
	// aussi: calculer et garder le déplacement x_disp (dit aussi
	// offset) du paramètre x dans le compilateur
	// et calculer la taille totale param_size des paramètres
	// appel de la fonction fun d'adresse fun
	LDW R0, #fun_ // charge adresse fun_ de fonction ds R1
	MPC WR // charge le contenu du PC dans WR
	ADQ 8, WR // WR contient l'adresse de retour
	ADQ -2, SP // décrémente le pointeur de pile SP
	STW WR, (SP) // sauve l'adresse de retour sur pile
	JEA (R0) // saute à l'instruction d'adresse ds R0
	// nettoyage paramètres
	LDW WR, #param_size // WR = taille totale des paramètres
	ADD WR, SP, SP // abandon paramètres

return <expression>	<code de calcul de l'expression, résultat dans R0>
	// UNLINK
	LDW SP, BP // abandon infos locales
	LDW BP, (SP) // charge BP avec ancien BP
	ADQ 2, SP // incrémente SP
	
	// retour au programme appelant
	LDW WR, (SP) // charge WR avec l'adresse de retour
	ADQ 2, SP // incrémente le pointeur de pile SP
	JEA (WR) // saute à instruction d'adresse ds WR

} **/*accolade fermante finale de main */**	LDW SP, BP // abandon infos locales
	LDW BP, (SP) // charge BP avec ancien BP
	ADQ 2, SP // incrémente SP
	
	LDW WR, #EXIT_EXC // WR = EXIT_EXC
	TRP WR // EXIT: arrête le programme
	
	// saute à main_ si redemande exécution
	LDW WR, #main_
	JEA (WR)

} **/*accolade fermante finale de fonction */**	LDW SP, BP // abandon infos locales
	LDW BP, (SP)+ // charge BP avec ancien BP
	
	// retour au programme appelant
	LDW WR, (SP) // charge WR avec l'adresse de retour
	ADQ 2, SP // incrémente le pointeur de pile SP
	JEA (WR) // saute à instruction d'adresse ds WR