2021/05/25 Naoki Hirayama

Vicopoid 仕様案

32bit の整数演算をする仮想コプロセッサ。8bit CPU でハッシュ関数を実装するのに便利なコプロセッサ。省メモリなどの効果がある。8bit CPU、WZeta、ICF3-Z の仮想マシン機能の動作検証が主題。

基本スペック

メモリ 1KB(256 ワード)
スタックポインタ SP 8bit
プロセッサと通信するためのポインタ P(WZeta では A:B)
インデックスレジスタ X 8bit (システム側の名前%VICO_X)
8bit の%R0〜%R11 までのワークレジスタを利用する。

スタックポインタ

スタックポインタの指すメモリはスタックトップのデータ。

実装について

省資源な仕様。ALU 命令は各ファンクションを個別にハードマクロ命令にすることで高速化が可能。

メモリの拡張

SHA-512 の演算では 64bit×80 個の定数が必要。32bit×256 個のメモリでも入る可能性はあるが、
もし入らない場合は別の 1KB のメモリを持つ PUSH 関数を追加して容量を拡張することが可能。

命令セット

n : 8bit の値　{}: 仮想コプロのメモリ　[]:プロセッサのメモリ %:プロセッサのメモリ

ニモニック	動作内容	長さ
PUSH n	{--%SP} = {n}	2
PUSH% %n	{--%SP} = {%n}	2
POP n	{n} = {%SP++}	2
POP% %n	{%n} = {%SP++}	2
PUSH8 n	{--%SP} = n	2
PUSH8% %n	{--%SP} = %n	2
SHIFT8% %n	{%SP} = {%SP}<<8 | n	2
ALU @ADD	{%SP+1} = {%SP+1}+{%SP}　　%SP += 1	2
ALU @SUB	{%SP+1} = {%SP+1}-{%SP}　　%SP += 1	2
ALU @MUL	{	2
ALU @AND	{%SP+1} = {%SP+1} &{%SP}　　%SP += 1	2
ALU @OR	{%SP+1} = {%SP+1}|{%SP}　　%SP += 1	2
ALU @XOR	{%SP+1} = {%SP+1}^{%SP}　　%SP += 1	2
ALU @NOT	{%SP}← !{%SP}	2
ALU @MEM	{%SP}←{{%SP}}	2
ALU @DUP	{%SP-1} = {%SP}　　%SP -= 1	2
ALU @TOP	%R3%R2%R1%R0 = {%SP}	2
SHL% %n	{%SP}<<= %n	2
SHR% %n	{%SP}>>= %n	2
READ n	[P] ← {n}	2
READ% %n	[P] ← {%n}	2
WRITE n	{n} ← [P]	2
MOV% %n	{X++} ← [P++]　(%n×4 バイトの転送)	2
MOVB% %n	{X++} ← [P++]　(%n バイトの転送)	2