TRAJAN - INSTRUCTION SET

{ Andreea.Chis, Matthieu.Gallet, Bogdan.Pasca } @ens-lyon.fr

*

Code	Name	Description
00000cccvvvvvvvv	(**LoadLo**)	charge les 8 bits v dans la partie basse de **ACC** et efface sa partie haute,
00001cccvvvvvvvv	(**LoadHi**)	charge les 8 bits v dans la partie haute de **ACC**,
00010ccc0iiiiiii	(**MoveAccToReg**)	recopie le contenu de **ACC** dans $\mathbf{R}_i$,
00011ccc0iiiiiii	(**MoveRegToAcc**)	recopie le contenu $\mathbf{R}_i$ dans **ACC**,
00100ccc0iiiiiii	(**Read**)	recopie le contenu de la mémoire à l'adresse **ACC** dans $\mathbf{R}_i$,
00101ccc0iiiiiii	(**Write**)	recopie le contenu de $\mathbf{R}_i$ dans la mémoire à l'adresse **ACC**,
00110ccc0iiiiiii	(**ReadInc**)	recopie le contenu de la mémoire à l'adresse **ACC** dans $\mathbf{R}_i$ et incrémente **ACC**,
00111ccc0iiiiiii	(**WriteInc**)	recopie le contenu de $\mathbf{R}_i$ dans la mémoire à l'adresse **ACC** et incrémente **ACC**,
01000ccc0iiiiiii	(**ReadDec**)	recopie le contenu de la mémoire à l'adresse **ACC** dans $\mathbf{R}_i$ et décrémente **ACC**,
01001ccc0iiiiiii	(**WriteDec**)	recopie le contenu de $\mathbf{R}_i$ dans la mémoire à l'adresse **ACC** et décrémente **ACC**,
01010ccc0iiiiiii	(**Jmp** – JuMP)	recopie le contenu de $\mathbf{R}_i$ dans **PC**,
01011ccc0iiiiiii	(**Jmr** – JuMp Relative)	ajoute le contenu (signé) de $\mathbf{R}_i$ à **PC**,
01100cccvvvvvvvv	(**Jmi** – JuMp Immediate)	charge les 8 bits v dans **PC**,
01101cccvvvvvvvv	(**Jmri** – JuMp Relative Immediate)	ajoute les 8 bits v (signés) à **PC**,
01110ccc0iiiiiii	(**Jsr** – Jump to SubRoutine)	recopie **PC** à l'adresse **SP**, incrémente **SP**, recopie le contenu de $\mathbf{R}_i$ dans **PC**,
01111cccvvvvvvvv	(**Jsri** – Jump to Subroutine Relative Immediate)	recopie **PC** à l'adresse **SP**, incrémente **SP**, ajoute les 8 bits v (signés) à **PC**,
10000ccc00000000	(**Rts** – ReTurn from Subroutine)	décrémente **SP**, recopie le contenu incrémenté de l'adresse **SP** dans **PC**
1000100000000000	(**Nop** – No OPeration)	no operation
10010cccsiiiiiii	(**Add**)	additionne le contenu de $\mathbf{R}_i$ à **ACC**,
10011cccsiiiiiii	(**Sub**)	soustrait le contenu de $\mathbf{R}_i$ à **ACC**,
10100cccsiiiiiii	(**Mul**)	multiplie les moins significatifs 8 bits de **ACC** par les moins significatifs 8 bits de $\mathbf{R}_i$ en 2's complement
10101cccsiiiiiii	(**Cmp** – CoMPare)	compare $\mathbf{R}_i$ à **ACC**,
10110cccsiiiiiii	(**Swap**)	échange les parties haute et basse de $\mathbf{R}_i$,
10111cccsiiiiiii	(**Clr** – CLeaR)	efface le contenu de $\mathbf{R}_i$,
11000cccsiiiiiii	(**And**)	copie le ET logique de **ACC** et $\mathbf{R}_i$ dans **ACC**,
11001cccsiiiiiii	(**Or**)	copie le OU logique de **ACC** et $\mathbf{R}_i$ dans **ACC**,
11010cccsiiiiiii	(**And**)	copie le XOU logique de **ACC** et $\mathbf{R}_i$ dans **ACC**,
11011cccsiiiiiii	(**Not**)	calcule le NON logique de $\mathbf{R}_i$ dans **ACC**,
11100cccsiiiiiii	(**Lsr** – Logical Shift Right)	décale **ACC** de $\mathbf{R}_i$ vers les bits de poids faible,
11101cccsiiiiiii	(**Lsl** – Logical Shift Left)	décale **ACC** de $\mathbf{R}_i$ vers les bits de poids fort,
11110cccsiiiiiii	(**Ror** – ROtate Right)	rotation de **ACC** de $\mathbf{R}_i$ vers les bits de poids faible,
11111cccsiiiiiii	(**RoL** – ROtate Left)	rotation de **ACC** de $\mathbf{R}_i$ vers les bits de poids fort.

*L'instruction est conditionné par les 3 bits ccc, et **SR** ne sera changé que si s est à 1.