[b]Immediate:[/b]

ADC.B Acc,#imm

[b]zero page:[/b]

LDB tmp,zp

ADC.B Acc,tmp

[b]zp[/b]

LDB tmp,zp

ADC.B Acc,tmp

[b]zp,x[/b]

LDB tmp,zp,x

ADC.B Acc,tmp

[b]zp,y[/b]

LDB tmp,zp,y

ADC.B Acc,tmp

[b] (zp,x)[/b]

LDW tmp,zp,x

LDB tmp,[tmp]

ADC Acc,tmp

[b](zp),y[/b]

LDW tmp,zp

ADD tmp,y

LDB tmp,[tmp]

ADC.B Acc,tmp

[b]abs[/b]

LDB tmp,abs

ADC Acc,tmp

[b]abs,x[/b]

LDB tmp,abs,x

ADC Acc,tmp

[b]abs,y[/b]

LDB tmp,abs,y

ADC Acc,tmp

LDB tmp,zp

LDB tmp,zp,x

LDB tmp,zp,y

LDB tmp,[tmp]

LDB tmp,abs

LDB tmp,abs,x

LDB tmp,abs,y

LDB tmp,sp ; for stack pulls

LDW tmp,zp

LDW tmp,zp,x

LDW tmp,[tmp] ; for JMP($abs)

MOV.B Acc,tmp

ADD.W tmp,x,#zp

ADD.W tmp,y

ADC.B Acc,#imm

ADC.B Acc,tmp

I’m toying with the idea of a superscalar 6502. It would work by changing 6502 opcodes into micro-ops in a manner similar to what’s done for the x86. So, I need to have worked out an appropriate set of micro-ops. The micro-ops would be a load / store architecture with a fixed 13-bit instruction format. I think all instructions can be implemented with a maximum of four micro-ops. This means a table of 54 bits for each instruction (2 bits used to indicate # of micro-ops). The table would be indexed by the opcode byte and a field (Ld3) in the micro-op instruction indicates when to take values from the macro-op instruction. Necessary for constants supplied by macro-ops.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Opcode6 | Ld4 | Rd3 | Rn3 |  | Ld4 | Meaning |
| NOP | ~ | ~ | ~ |  |  |  |
| LDB | Ld3 | Rd3 | Rn2 |  | 0 | no constant |
| LDW | Ld3 | Rd3 |  |  | 1 | the value 1 |
| STB | Ld3 | Rs3 |  |  | 2 | the value 2 |
| ADD.W | Ld3 | Rd3 |  |  | 3 | the value3 |
| ADD.B | Ld3 | Rd3 |  |  | 4 |  |
| ADC.B | Ld3 | Rd3 |  |  | 5 |  |
| SBC.B | Ld3 | Rd3 |  |  | 6 |  |
| AND.B | Ld3 | Rd3 |  |  | 7 | the value $100 |
| OR.B | Ld3 | Rd3 |  |  | 8 | reference 8 bits from macro op (byte 2) |
| EOR.B | Ld3 | Rd3 |  |  | 9 | reference 16 bits from macro op (bytes 2,3) |
| CMP.B | Ld3 | Rd3 |  |  | 10 |  |
| ADC.B tmp | Ld3 | Rd3 |  |  | 11 |  |
| SBC.B tmp | Ld3 | Rd3 |  |  | 12 |  |
| AND.B tmp | Ld3 | Rd3 |  |  | 13 | the value -3 |
| OR.B tmp | Ld3 | Rd3 |  |  | 14 | the value -2 |
| EOR.B tmp | Ld3 | Rd3 |  |  | 15 | the value -1 |
| CMP.B tmp | Ld3 | Rd3 |  |  |  |  |
| BEQ | 4 | ~ |  |  |  |  |
| BNE | 4 | ~ |  |  |  |  |
| BCS | 4 | ~ |  |  |  |  |
| BCC | 4 | ~ |  |  |  |  |
| BVS | 4 | ~ |  |  |  |  |
| BVC | 4 | ~ |  |  |  |  |
| BMI | 4 | ~ |  |  |  |  |
| BPL | 4 | ~ |  |  |  |  |
| CLC | ~ | ~ |  |  |  |  |
| SEC | ~ | ~ |  |  |  |  |
| CLV | ~ | ~ |  |  |  |  |
| SEI | ~ | ~ |  |  |  |  |
| CLI | ~ | ~ |  |  |  |  |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Rd3 | Register |  | Rn3 | Index Register |
| 0 | Acc |  | 0 | z (none) |
| 1 | x |  | 1 | x |
| 2 | y |  | 2 | y |
| 3 | sp |  | 3 | sp |
| 4 | pc |  |  |  |
| 5 | tmp |  | 5 | tmp |
| 6 | pc+2 |  |  |  |
| 7 | sr |  |  |  |

Some sample instruction breakdowns:

[code]

[b]pha[/b]

SB acc,sp

ADD.B sp,#-1

[b]adc (zp),y[/b]

LDW tmp,zp

ADD tmp,y

LDB tmp,[tmp]

ADC.B Acc,tmp

[b]rti[/b]

ADD.B sp,#1

LDB sr,sp

ADD.B sp #2

LDW pc,sp-1

[b]rts[/b]

ADD.B sp,#2

LDW pc,sp-1

ADD.w pc,#1

[/code]