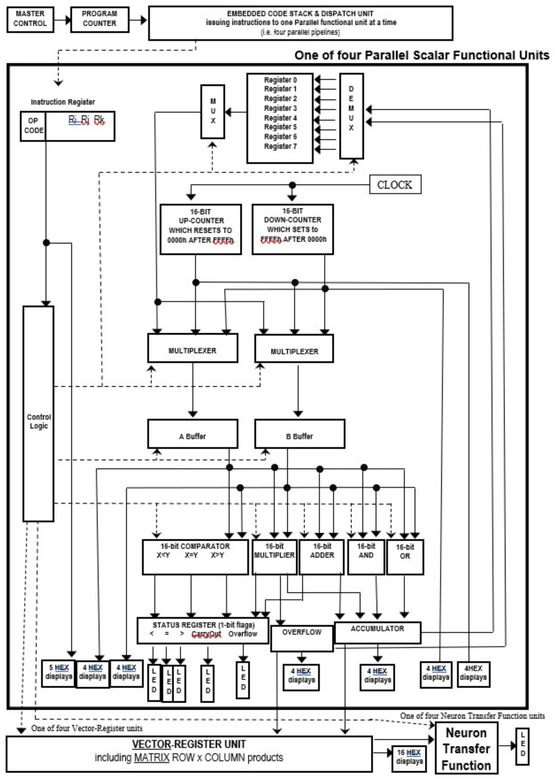
4-Way Superscalar and Vector-Register Processor Design

Joseph Kutteh, Nica McDevitt, Nathan Griffin

ETOWN
ELIZABETHTOWN COLLEGE

DESIGN SPECIFICATIONS



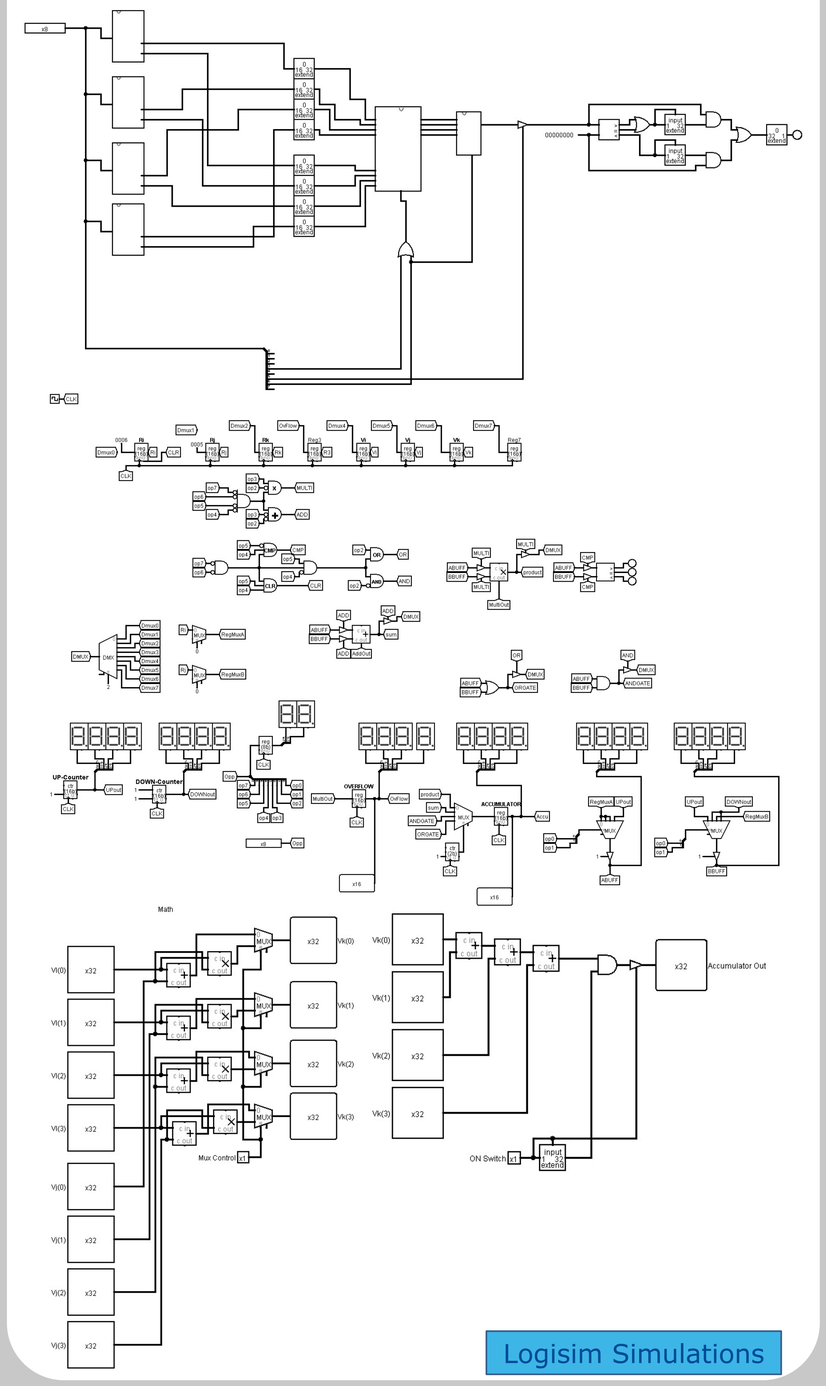
REFERENCES

[1] J. Freaney, “8051 Simulator,” 12-May-2019. [Online]. Available: http://users.etown.edu/w/wunderjt/LAB%20MANUALS%20post-2018/2019%20MANUAL%208051Microcontroller%20Simulator.pdf.

[2] J. Edmonds and J. Wunderlich, “2022 LECTURE: Intel 8051 Microcontroller Simulation Tutorial, by Jeff Edmonds,” *YouTube*, 17-Apr-2022. [Online]. Available: https://www.youtube.com/watch?v=KyG2FSVZ7Fk&list=PLK3MJsXEYEQIGw3tmIkjsBfZ49rr_GAmv&index=9&t=16s.

[3] J. T. Wunderlich PhD, “LAB #5-6 4-WAY SUPERSCALAR + VECTOR-REGISTER PROCESSOR DESIGN with Parallel Neuron Transfer Functions Circuit Simulations, Assembly language, FPGA-w/HDL, TTL Chips, and Dual PLC’s,” 13-Apr-2022.

TECHNICAL DESIGN



Logisim Simulations

TESTING

Machine Instruction Set

SCALAR ARITHMETIC ADDITION
- 00h (OP-CODE = 00000000) Ri + counter#1 →Rk
- 01h (OP-CODE = 00000001) Ri + counter#2 →Rk
- 02h (OP-CODE = 00000010) Ri + Rj →Rk
- 03h (OP-CODE = 00000011) counter#1 + counter#2 →Rk

SCALAR ARITHMETIC SUBTRACTION
- 04h to 07h (OP-CODE = 000001XX) *Reserved for future subtraction instructions*

SCALAR ARITHMETIC MULTIPLICATION
- 08h (OP-CODE = 00001000) Ri x counter#1 →Rk, overflow →R(k+1)
- 09h (OP-CODE = 00001001) Ri x counter#2 →Rk, overflow →R(k+1)
- 0Ah (OP-CODE = 00001010) Ri x Rj →Rk, overflow →R(k+1)
- 0Bh (OP-CODE = 00001011) counter#1 x counter#2 →Rk, overflow→R(k+1)

SCALAR ARITHMETIC DIVISION
- 0Ch to 0Fh (OP-CODE = 000011XX) *Reserved for future division instructions*

SCALAR ARITHMETIC COMPARISON
- 10h (OP-CODE = 00010000) Compare Ri with counter#1 →Rk
- 11h (OP-CODE = 00010001) Compare Ri with counter#2 →Rk
- 12h (OP-CODE = 00010010) Compare Ri with Rj →Rk
- 13h (OP-CODE = 00010011) Compare Counters

SCALAR LOGICAL AND
- 20h (OP-CODE = 00100000) Ri AND counter#1 →Rk
- 21h (OP-CODE = 00100001) Ri AND counter#2 →Rk
- 22h (OP-CODE = 00100010) Ri AND Rj →Rk
- 23h (OP-CODE = 00100011) AND counters →Rk

SCALAR LOGICAL OR
- 24h (OP-CODE = 00100100) Ri OR counter#1 →Rk
- 25h (OP-CODE = 00100101) Ri OR counter#2 →Rk
- 26h (OP-CODE = 00100110) Ri OR Rj →Rk
- 27h (OP-CODE = 00100111) OR counters →Rk

CLEAR
- 30h (OP-CODE = 00110000) Clear Ri

Vector/Array/Matrix and Neuron Instructions

VECTOR ARITHMETIC ADDITION
- 82h (OP-CODE = 10000010) Vi + Vj →Vk

VECTOR ARITHMETIC SUBTRACTION
- 84h to 87h (OP-CODE = 100001XX) *Reserved for future subtraction instructions*

VECTOR ARITHMETIC MULTIPLICATION
- 0Ah (OP-CODE = 10001010) Vi x Vj →Vk, overflow →V(k+1)

VECTOR ARITHMETIC DIVISION
- 8Ch to 8Fh (OP-CODE = 100011XX) *Reserved for future division instructions*

MATRIX ROW x COLUMN (i.e., Dot-Product)
- C0h (OP-CODE = 11000000) Vi x Vj →Vk, overflow →V(k+1)
  Vk(1)+Vk(2)+Vk(3)+Vk(4) → 32-Bit Scalar Accumulator

NEURON TRANSFER FUNCTION
- E0h (OP-CODE = 11100000) Vi x Vj →Vk, overflow →V(k+1)
  Vk(1)+Vk(2)+Vk(3)+Vk(4) → 32-Bit Scalar Accumulator
  32-Bit Scalar Accumulator → **Neuron Transfer Function**

8051 Simulations

```
;moving max and min values, 15 due to 4 bits
MOV 20h, #15
MOV 21h, #0
MOV 22h, #15
main:
        MOV P1, P2
        JMP rowScan
        JMP main
rowScan:
        CLR P0.3
        JMP colScan
colScan:
        JNB P0.6, gotKey1
        JNB P0.5, gotKey2
        JMP main
;check if 1 on keypad is pressed
gotKey1:
        MOV A, 20h
        JZ rstKey1
        DEC 20h
        JMP inputVal
;check if 2 on keypad is pressed
gotKey2:
        MOV A, 22h
        SUBB A, 21h
        JZ rstKey2
        INC 21h
        JMP inputVal
rstKey1:
        MOV 20h, #15
        RET
rstKey2:
        MOV 21h, #0
        RET
;check which switch is pressed 0-7
inputVal:
        CLR A
        JNB P2.6, sw6
        JNB P2.5, sw5
        JNB P2.4, sw4
        JNB P2.3, sw3
;if switch 6 is pressed
sw6:
        MOV A, #8
        JNB P2.5, sw5
        JNB P2.4, sw4
        JNB P2.3, sw3
        MOV R0, A
        JNB P1.7, chkUp
        JB P1.7, chkDn
;if swicth 5 is pressed
sw5:
        ADD A, #4
        JNB P2.4, sw4
        JNB P2.3, sw3
        MOV R0, A
        JNB P1.7, chkUp
        JB P1.7, chkDn
;if swicth 4 is pressed
sw4:
        ADD A, #2
        JNB P2.3, sw3
        MOV R0, A
        JNB P1.7, chkUp
        JB P1.7, chkDn

;if swicth 3 is pressed
sw3:
        ADD A, #1
        MOV R0, A
        JNB P1.7, chkUp
        JB P1.7, chkDn
chkUp:
        CJNE A, 21h, comp
        CLR P1.1
        JMP delay
chkDn:
        CJNE A, 20h, comp
        CLR P1.1
        JMP delay
comp:
        JNC compGreat
        JC compLess
compLess:
        CLR P1.2
        JMP delay
compGreat:
        CLR P1.0
        JMP delay
delay:
        MOV R1, #20
        DJNZ R1, $
        JMP main
```