به نام خداوند بخشنده مهربان

آرزو پاکزاد
۹۵۱۰۵۴۳۲

---

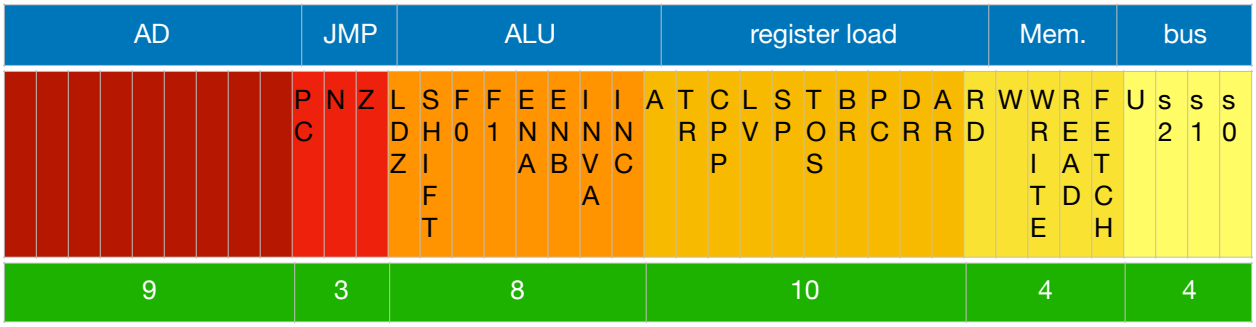
label	RTL
BIPUSH	1: AR = SP + 1, SP = SP + 1, fetch 2: BR = M[PC] 3: DR = BR, TOS = BR 4: PC = PC + 1, write 5: M[AR] = DR, w
GOTO	1: TR = PC - 1 2: PC = PC + 1, fetch 3: BR = M[PC] 4: A = BR << 8, fetch 5: BR = M[PC] 6: A = A OR BR(unsigned) 7: PC = TR + A
IADD	1: AR = SP - 1, SP = SP - 1 2: A = TOS, read 3: DR = M[AR], rd 4: DR = A + DR, TOS = A + DR 5:write 6: M[AR] = DR, w
TRUE	1:TR = PC - 1, jmp goto2
FALSE	1: PC = PC + 1 2: PC = PC + 1
IFEQ	1: AR = SP - 1, SP = SP - 1 2: TR = TOS, read 3: DR = M[AR], rd 4: TOS = DR 5: Z = TR, if(Z) jmp TRUE else jmp FALSE
IFLT	T1: AR = SP - 1, SP = SP - 1 T2: TR = TOS, read T3: DR = M[AR], rd T4: TOS = DR T5: N = TR, if(N) jmp TRUE else jmp FALSE
IF_ICMPEQ	T1: AR = SP - 1, SP = SP - 1 T2: AR = SP - 1, SP = SP - 1, read T3: DR = M[AR] , TR = TOS, rd T4: A = DR, read T5: DR = M[AR], rd T6: TOS = DR T7: Z = TR - A, if(Z) jmp TRUE else jmp FALSE

label	RTL
IINC	1: PC = PC + 1, fetch 2: BR = M[PC], A = LV 3: AR = A + BR(unsigned), fetch 4: BR = M[PC], read 5: DR = M[AR], rd 6: A = DR 7: DR = A + BR 8: PC = PC + 1, write 9: M[AR] = DR, w
ILOAD	1: A = LV, fetch 2: BR = M[PC] 3: AR = A + BR(unsigned) 4: AR = SP + 1, SP = SP + 1, read 5: DR = M[AR], rd 6: PC = PC + 1, write 7: M[AR] = DR, TOS = DR, w
ISTORE	1: A = LV, fetch 2: BR = M[PC] 3: DR = TOS 4: AR = A + BR(unsigned) 5: AR = SP -1, SP = SP - 1, write 6: M[AR] = DR, w 7: PC = PC + 1, read 8: DR = M[AR], rd 9: TOS = DR
ISUB	1: AR = SP - 1, SP = SP - 1 2: A = TOS, read 3: DR = M[AR], rd 4: DR = DR - A, TOS = DR - A 5:write 6: M[AR] = DR,w
NOP	1:
FETCH	1: fetch 2: BR = M[PC] 3: PC = PC + 1, jmp BR

label	RTL
DUP	1: AR = SP + 1, SP = SP + 1, read 2: DR = TOS, rd 3: write 4: M[AR] = DR, w
IAND	1: AR = SP - 1, SP = SP - 1 2: A = TOS, read 3: DR = M[AR], rd 4: DR = A AND DR, TOS = A AND DR 5: write 6: M[AR] = DR, w

label	RTL
IOR	1: AR = SP - 1, SP = SP - 1 2: A = TOS, read 3: DR = M[AR], rd 4: DR = A OR DR, TOS = A OR DR 5: write 6: M[AR] = DR, w
SWAP	1: AR = SP - 1, read 2: DR = M[AR], AR = SP, rd 3: TR = DR, write 4: M[AR] = DR, DR = TOS, w 5: AR = SP - 1 6: TOS = TR, write 7: M[AR] = DR, w
LDC_W	1: PC = PC + 1, fetch 2: BR = M[PC] 3: A = BR(unsigned) << 8, fetch 4: BR = M[PC] 5: A = A OR BR 6: AR = A + CPP 7: AR = SP + 1, SP = SP + 1, read 8: DR = M[AR], rd 9: PC = PC + 1, write 10: M[AR] = DR, TOS = DR, w
IRETURN	1: AR = LV, SP = LV 2: read 3: DR = M[AR], rd 4: LV = DR, AR = DR 5: read 6: DR = M[AR], rd 7: AR = LV + 1 8: PC = DR 9: DR = M[AR], AR = SP 10: LV = DR 11: DR = TOS 12: write 13: M[AR] = DR, w
POP	1: AR = SP - 1, SP = SP - 1 2: read 3: DR = M[AR], rd 4: TOS = DR
WIDE	1: PC = PC + 1 2: BR = M[PC], jmp BR OR (0x100)
WIDE_ILOAD	1:PC = PC + 1, fetch 2:BR = M[PC] 3:A = BR << 8, fetch 4:BR = M[PC] 5:A = A OR BR(unsigned) 6:AR = LV + A, jmp ILOAD4

label	RTL
WIDE_ISTORE	1:PC = PC + 1, fetch 2:BR = M[PC] 3:A = BR << 8, fetch 4:BR = M[PC] 5:A = A OR BR(unsigned) 6:AR = LV + A, jmp ISTORE5
WIDE_IINC	1:PC = PC + 1, fetch 2:BR = M[PC] 3:A = BR << 8 4:PC = PC + 1, fetch 5:BR = M[PC] 6:A = A OR BR(unsigned) 7:AR = A + LV, jmp IINC4
INVOKEVIRTUAL	PC = PC + 1, fetch BR = M[PC] A = BR << 8, fetch BR = M[PC] A = A OR BR(unsigned) AR = A + CPP TR = PC + 1, read DR = M[AR], rd PC = DR PC = PC + 1, fetch BR = M[PC] A = BR << 8, fetch BR = M[PC] PC = PC + 1 A = A OR BR(unsigned), fetch BR = M[PC] TOS = SP - A TOS = TOS + 1, AR = TOS + 1 PC = PC + 1, fetch BR = M[PC] A = BR << 8, fetch BR = M[PC] A = A OR BR(unsigned) DR = SP + A + 1 AR = DR, SP = DR, write M[AR] = DR, w DR = TR PC = PC + 1, write M[AR] = DR, w AR = SP + 1, SP = SP + 1 DR = LV, fetch BR = M[PC], write M[AR] = DR, LV = TOS, w



write: یک کلاک قبل از نوشتن از DR در M[AR]

read: یک کلاک قبل از خواندن از M[AR] در DR

rd: در حین خواندن از M[AR] در DR

w: در حین نوشتن از DR در M[AR]

fetch: یک کلاک قبل از خواندن از M[PC] در BR

کد زیر در مموری نوشته شده است:

wide
iload
0
1
bipush
0
bipush
2
bipush
1
invokevirtual
0
1
iand
istore
3

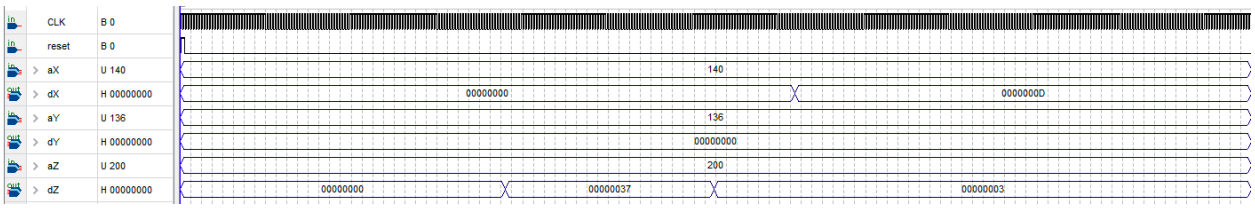
تابع صدازده شده:

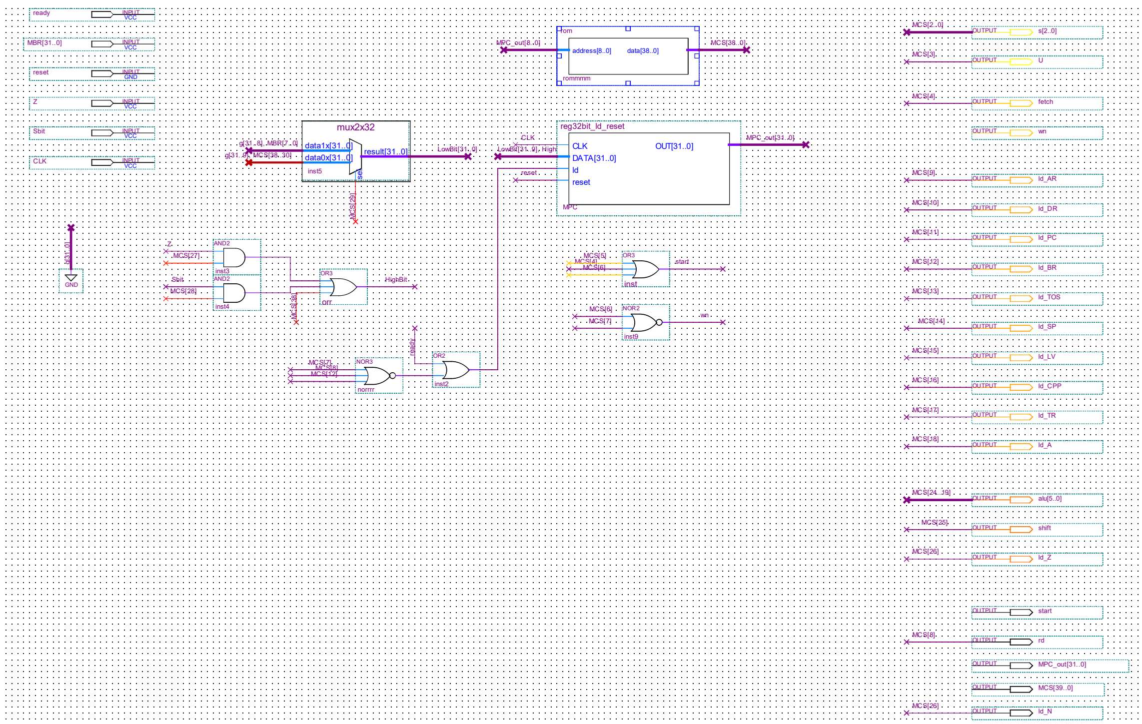
0
3

0
2
iload
1
iload
2
ior
ireturn

```
//PC
array[0] <= 8'hC4;
array[1] <= 8'h15;
array[2] <= 8'h00;
array[3] <= 8'h01;
array[4] <= 8'h10;
array[5] <= 8'h00;
array[6] <= 8'h10;
array[7] <= 8'h02;
array[8] <= 8'h10;
array[9] <= 8'h01;
array[10] <= 8'hB6;
array[11] <= 8'h00;
array[12] <= 8'h01;
array[13] <= 8'h60;
array[14] <= 8'h36;
array[15] <= 8'h03;

array[20] <= 8'h00;
array[21] <= 8'h03;
array[22] <= 8'h00;
array[23] <= 8'h02;
array[24] <= 8'h15;
array[25] <= 8'h01;
array[26] <= 8'h15;
array[27] <= 8'h02;
array[28] <= 8'h80;
array[29] <= 8'hAC;

//CPP
array[64] <= 8'h14;
array[68] <= 8'h14;

//LV
array[128] <= 8'h01;
array[132] <= 8'h0A;

//SP
array[192] <= 8'h01;
```

waveform:




ready
MBR[31..0]
reset
Z
Sbit
CLK
mux2x32
data1x[31..0]
data0x[31..0]
result[31..0]
inst5
rom
address[8..0]
data[38..0]
rommmm
reg32bit_ld_reset
CLK
DATA[31..0]
ld
reset
OUT[31..0]
MPC
MPC_out[8..0]
MCS[38..0]
MPC_out[31..0]
GND
HighBit
start
wn
s[2..0]
U
fetch
wn
ld_AR
ld_DR
ld_PC
ld_BR
ld_TOS
ld_SP
ld_LV
ld_CPP
ld_TR
ld_A
alu[5..0]
shift
ld_Z
start
rd
MPC_out[31..0]
MCS[38..0]
ld_N