Bits index	What bits represent
15	Instruction words count
14	ALU out write back en
13	memory en – ALU out WB (dst/src)
12	Data out write back en
11	Rd / Wr
10	Push PC
9 : 6	ALU operation
5 : 3	Src
2 : 0	Dst

Special Cases:

1- Mul : write back high ALU out to Src
2- LDD: bits 13 and 11 : 3 contain effective address so control signals are implicit
3- STD: bits 13 and 11 : 3 contain effective address so control signals are implicit
4- Bits 2 : 0 = 7 means out.port
5- Bits 5 : 3 = 7 means in.port
6- ALU operation may specified by the means of the rest of instruction
7- Memory write address = mar, read address = ALU out
8- In ALU branch is another code for dec

Notes

1- Processor doesn’t support nested interrupts
2- Processor is in complete freeze while pipeline is stalled (doesn’t respond to interrupt signal until pipeline returns to ordinary state)
3- Load case doesn’t cause data hazards because data memory writes data in rising edge (data is passes at once to execution stage, decode stage and to fetch stage(return case fetch needs loaded PC))
4- Stalling happens due to control hazards and in return case
5- Interrupt pushed an instruction in the pipeline and this instruction is executed normally
6- R6 is SP
7- PC is not included in register file fetch stage only has control on it
8- Intr and reset signals should be raised for specified interval
9- Clk sync are included in design and forced in do files

Static Prediction: Not taken
pipeline hazards solved by complete forwarding in decode and branch stage

```
-- flag bit 0: carry
--                bit 1: zero
--                bit 2: negative
-- 0000 F = A
-- 0001 F = A + 1
-- 0010 F = A + B
-- 0011 F = A - B
-- 0100 F = A - 1

-- 0101 F = A and B
-- 0110 F = A or B
-- 0111 F = not A

-- 1000 F = A << B
-- 1001 F = A >> B
-- 1010 F = RLC A
-- 1011 F = RRC A

-- 1100 F = Branch
-- 1101 F = Call
-- 1110 F = Carry
-- 1111 F = A * B
```

Opcodes

instr	15	14	13	12	11	10	9 : 6	5 : 3	2 : 0
nop	0	0	0	0	0	0	0	0	0
out	0	0	0	0	0	0	F = A	Rdst	7
clrc	0	0	0	0	0	0	carry	0	2
setc	0	0	0	0	0	0	carry	0	3
jmp	0	0	0	0	0	0	branch	0	Rdst
jn	0	0	0	0	0	0	branch	1	Rdst
jc	0	0	0	0	0	0	branch	2	Rdst
jz	0	0	0	0	0	0	branch	4	Rdst
in	0	1	0	0	0	0	F = A	7	Rdst
mov	0	1	0	0	0	0	F = A	Rsrc	Rdst
add	0	1	0	0	0	0	A + B	Rsrc	Rdst
mul	0	1	0	0	0	0	A * B	Rsrc	Rdst
sub	0	1	0	0	0	0	A - B	Rsrc	Rdst
and	0	1	0	0	0	0	A & B	Rsrc	Rdst
or	0	1	0	0	0	0	A | B	Rsrc	Rdst
rlc	0	1	0	0	0	0	F = rlcA	Rsrc	Rdst
rrc	0	1	0	0	0	0	F = rrcA	Rsrc	Rdst
not	0	1	0	0	0	0	F = ~A	Rsrc	Rdst
inc	0	1	0	0	0	0	Inc A	Rsrc	Rdst
dec	0	1	0	0	0	0	Dec A	Rsrc	Rdst
shl	0	1	Imm	Imm	Imm	Imm	F = shlA	Rsrc	Rdst
shr	0	1	Imm	Imm	Imm	Imm	F = shrA	Rsrc	Rdst
ret	0	1	1	0	0	0	Inc A	6	0
rti	0	1	1	0	0	0	Inc A	6	1
push	0	1	1	0	1	0	Dec A	6	Rdst
call	0	1	1	0	1	1	call	6	Rdst
interrupt	0	1	1	0	1	1	Dec A	6	0
pop	0	1	1	1	0	0	Inc A	6	Rdst
std	1	0	EA	0	EA	EA	EA	EA	Rsrc
ldd	1	0	EA	1	EA	EA	EA	EA	Rdst
ldm	1	1	0	0	0	0	F = A	0	Rdst