Control Signals

PC_{Save}	[1-bit]
Branch	[1-bit]
BranchCondition	[1-bit]
Flush	[1-bit]
Stall	[1-bit]
SrcData/MemData	[16-bits]
DstData/MemAddr/ShiftAmount	[16-bits]
Rsrc	[3-bits]
RsrcWB	[1-bit]
RsrcLoad	[1-bit]
Rdst	[3-bits]
RdstWB	[1-bit]
RdstLoad	[1-bit]
ImmediateLoad	[1-bit]
MOV	[1-bit]
ALU_{Opr}	[4-bits]
$Flag_{En}$	[1-bit]
$Flag_{Restore}$	[1-bit]
MemAddr	[10-bits]
$MemAddr_{Switch}$	[1-bit]
MemRD	[1-bit]
MemWR	[1-bit]
PortInRD	[1-bit]
PortOutWR	[1-bit]

Data Forwarding

During Decode Stage:

PC_{i+1}	=	PC_Reset	When	Reset
		PC_INTR	When	INTR
		WB_SrcData	When	WB_Rsrc=PC AND WB_RsrcWB
		DEC_SrcData	When	BranchTaken
		PC_i+1	Otherwise	
SrcData	=	$Instr_{i+1}$	When	ImmediateLoad
		Flags & PC_{i+1}	When	PC_{Save}
		DstData	When	$MemAddr_{Switch}$
		MEM_SrcData	When	DEC_Rsrc=MEM_Rsrc AND MEM_RsrcWB
		MEM_DstData	When	DEC_Rsrc=MEM_Rdst AND MEM_RdstWB
		WB_SrcData	When	DEC_Rsrc=WB_Rsrc AND WB_RsrcWB
		WB_DstData	When	DEC_Rsrc=WB_Rdst AND WB_RdstWB
		R[DEC_Rsrc]	Otherwise	
DstData	=	IN_Port	When	PortInRD
		EA	When	Load/Store
		ShiftAmount	When	ALU_{Opr}=Shift
		MEM_SrcData	When	DEC_Rdst=MEM_Rsrc AND MEM_RsrcWB
		MEM_DstData	When	DEC_Rdst=MEM_Rdst AND MEM_RdstWB
		WB_SrcData	When	DEC_Rdst=WB_Rsrc AND WB_RsrcWB
		WB_DstData	When	DEC_Rdst=WB_Rdst AND WB_RdstWB
		R[DEC_Rdst]	Otherwise	

During Memory Stage:

$SrcData_{i+1}$	=	MemDout	When	MEM_MemRD
		$WB\_SrcData_i$	When	MEM_Rsrc=WB_Rsrc AND WB_MemRD
		MEM_SrcData	Otherwise	

Stall & Flush

Stall:

1. Load/Pop Use [1 stall]

```
Stall =  EXE_MemRD AND (
             (EXE_Rsrc=DEC_Rsrc AND DEC_RsrcLoad) OR
             (EXE_Rsrc=DEC_Rdst AND DEC_RdstLoad)
         )
```

- Disable IR & PC (to keep the instruction).
- Reset DEC/EXE buffers.

Flush:

1. Load Immediate [1 flush]
2. Branch Taken [1 flush]
3. RET/RTI [3 flushes]

```
Flush =  ImmediateLoad                    OR
         BranchTaken                      OR
         (DEC_Rsrc=PC AND DEC_RsrcWB)     OR
         (EXE_Rsrc=PC AND EXE_RsrcWB)     OR
         (MEM_Rsrc=PC AND MEM_RsrcWB)     OR
         (WRB_Rsrc=PC AND WRB_RsrcWB)
```

- Reset the IR (i.e. convert the instruction into bubble).

Stage Buffers

SRC Buffer:

WB	Rsrc	SrcData
1	3-bits	16-bits

WB: Write Back in Rsrc

DST Buffer:

WB	Rdst	DstData
1	3-bits	16-bits

WB: Rdst Write Back

CTRL Buffer:

ALU_{Opr}	FE	FR	AS	WR	RD	OP
4-bits	6-bits					

FE: Flags Write Enable
FR: Flags Restore Signal
AS: Memory Address Switch
When AS=1 take address from DstData, otherwise from MemAddr.
WR: Memory Write
RD: Memory Read
OP: Output SrcData to OUT port

ALU specs

Inputs: Src, Dst
Outputs: Res1, Res2, Flags

0-operand ALU operations	
NOP	00 00
MOV	00 01
CLRC	00 10
SETC	00 11
Add/Sub ALU operations	
ADD	01 00
SUB	01 01
INC	01 10
DEC	01 11
Mul/Logic ALU operations	
MUL	10 00
AND	10 01
OR	10 10
NOT	10 11
Shift/Rotate ALU operations	
RLC	11 00
RRC	11 01
SHL	11 10
SHR	11 11

Move instructions

0	0	WB	MV	LM	IP	OP	-	-	-	Rdst	Rsrc
1	1	8-bits								3-bits	3-bits

Signals

WB: Write Back in Rdst
MV: Move SrcData to DstData in Memory Stage
LM: Load Immediate
IP: Port In
OP: Port Out

Opcodes

Instr	**00**	**WB**	**MV**	**LM**	**IP**	**OP**	**-**	**Rdst**	**Rsrc**
NOP		0	0	0	0	0	-	-	-
MOV Rsrc, Rdst		1	1	0	0	0	-	Rdst	Rsrc
LDM R, #		1	0	1	0	0	-	R	-
IN R		1	0	0	1	0	-	R	-
OUT R		0	0	0	0	1	-	-	R

#: Immediate value in next address of the Program Memory.

RsrcWB = 0, RsrcLoad = MV
RdstWB = WB, RdstLoad = 0

ALU instructions

0	1	ALU_{Opr}	Imm	Rdst	Rsrc
1	1	4-bits	4-bits	3-bits	3-bits

Opcodes

Instr	**01**	**ALU_{Opr}**	**Imm**	**Rdst**	**Rsrc**
SETC		00 11		-	-
CLRC		00 10		-	-
SUB Rsrc, Rdst		01 01		Rdst	Rsrc
INC Rdst		01 10		Rdst	Rdst
DEC Rdst		01 11		Rdst	Rdst
MUL Rsrc, Rdst		10 00		Rdst	Rsrc
AND Rsrc, Rdst		10 01		Rdst	Rsrc
OR Rsrc, Rdst		10 10		Rdst	Rsrc
NOT Rdst		10 11		Rdst	Rdst
RLC Rdst		11 00		Rdst	Rdst
RRC Rdst		11 01		Rdst	Rdst
SHL Rsrc, #, Rdst		11 10	#	Rdst	Rsrc
SHR Rsrc, #, Rdst		11 11	#	Rdst	Rsrc

RsrcWB = MUL
RsrcLoad = (ALU_{Opr} != Type 0)

RdstWB = (ALU_{Opr} != Type 0)
RdstLoad = (ALU_{Opr} != Type 0) AND (ALU_{Opr} != Shift Opr)

FlagsEN = 1

Memory instructions

1	0	LD	Effective Address (EA)	Rsrc
1	1	1	10-bits	3-bits

Signals

LD: Load/Store Flag bit

Opcodes

Instr	**10**	**LD**	**EA**	**Rsrc**
LDD R, EA		1	EA	R
STD R, EA		0	EA	R

EA: Effective Address

RsrcWB = LD, RsrcLoad = 0
RdstWB = 0, RdstLoad = 0, DstData = EA

MemRD = LD, MemWR = !LD, $MemAddr_{Switch}$ = 0

Branch & Stack instructions

1	1	BR_{Switch}	BR	ST	PP	FR	SP	TR	Rdst	Rsrc
1	1	2-bits	6-bits						3-bits	3-bits

Signals

BR_{Switch}: Branch Switch
BR: Branch
ST: Stack operation
PP: Pop from stack flag bit if ST=1
FR: Flags Restore
SP: Save PC & Flags
TR: Interrupt

Opcodes

Instr	**11**	**BR_{Cond}**	**BR**	**ST**	**PP**	**FR**	**SP**	**TR**	**Rdst**	**Rsrc**
JMP R		00	1	0					-	R
JZ R		01	1	0					-	R
JN R		10	1	0					-	R
JC R		11	1	0					-	R
CALL R		00	1	1	0		1		SP	R
RET		-	0	1	1				SP	PC
RTI		-	0	1	1	1			SP	PC
PUSH R		-	0	1	0				SP	R
POP R		-	0	1	1				SP	R
INTR		-	0	1	0		1	1	SP	-

RsrcWB = Pop, RsrcLoad = Branch
RdstWB = Stack, RdstLoad = Stack
ALU_{Opr} = INC When Pop, DEC When Push, NOP Otherwise

MemRD = Pop, MemWR = Push, $MemAddr_{Switch}$ = Pop