1. Fetch
2. Decode

# Generated Signals:

## Signal S:

A signal to detect if the two instructions can be executed together, or an instruction needs two memory words and the second part of it hasn’t been fetched yet (LDM).

S = ‘1’ if   
(dst1 = dst2) && (dst1, dst2 exist) && (inst2!= {IN, POP,LDD}) OR  
(dst1 = src2) && (src2, dst1 exist) OR  
(inst2 = LDM) OR  
(inst1 && inst2 use memory ) OR  
(inst1 = (oneOp || twoOp) && not(NOP,OUT,MOV) )&& (inst2 = branch)

## Signal WB1, WB2:

WB1 = dst1 && (inst1 != STD)  
WB2 = dst2 && (ints2 != STD)

## Signal W/R Memory:

W = (inst1 or inst2 = {PUSH, STD,CALL })  
R = (inst1 or inst2 = {POP, LLD, RETURN })

## First Operand

|  |  |
| --- | --- |
| **Instruction** | **ALU** |
| NOP | F=0 |
| SETC | C <= 1 |
| CLRC | C <= 0 |
| NOT | F = NOT B |
| INC | F = INC B |
| DEC | F = DEC B |
| OUT | F = A |
| IN | F=0 |

## Two Operands

|  |  |
| --- | --- |
| **Instruction** | **ALU** |
| MOV | F = A |
| ADD | F = A + B |
| SUB | F = B – A |
| AND | F = A and B |
| OR | F= A OR B |
| SHL | F = SHL B |
| SHR | F = SHR B |

## Memory

|  |  |
| --- | --- |
| **Instruction** | **ALU** |
| STD | F = A |
| Others | F=0 |

## Branch

|  |  |
| --- | --- |
| **Instruction** | **ALU** |
| JZ , JN , JC , CALL , JUMP | F = B |
| Others | F=0 |

1. Execute

# Generated Signals:

## Bubble:

A signal to indicate load-use.

B = ‘1’ if:  
(ID/IE.R1) && (ID/IE.dst1 = ( (IF/ID.dst1 && IF/ID.dst1 exist) ||   
(IF/ID.dst2 && IF/ID.dst2 exist ) || ( IF/ID.src1 && IF/ID.src1 exist ) ||  
 (IF/ID.src2 && IF/ID.src2 exist ) ) ) OR  
(ID/IE.R2) && (ID/IE.dst2 = ( (IF/ID.dst1 && IF/ID.dst1 exist) ||   
(IF/ID.dst2 && IF/ID.dst2 exist ) || ( IF/ID.src1 && IF/ID.src1 exist ) ||  
 (IF/ID.src2 && IF/ID.src2 exist ) ) ).

## Jump:

Jmp1 = ‘1’ if  
(C = ‘1’ && inst1= JC) || (N=’1’ && inst1 = JN) || (Z=’1’ && inst1 = JZ) || (.inst1 = JMP) || (inst1 = CALL)

Jmp2 = ‘1’ if  
(C = ‘1’ && inst2= JC) || (N=’1’ && inst2 = JN) || (Z=’1’ && inst2 = JZ) || (inst2 = JMP) || (inst2 = CALL)

## Memory Forward:

MF = A1 || B1 || C1 || D1 || A2 || B2 || C2 || D2

A1 = (IM/WB.WB1) && (IM/WB.dst1 = (ID/IE.dst1 && ID/IE.dst1 exist))  
B1 = (IM/WB.WB1) && (IM/WB.dst1 = (ID/IE.dst2 && ID/IE.dst2 exist ))  
C1 = (IM/WB.WB1) && (IM/WB.dst1 = ( ID/IE.src1 && ID/IE.src1 exist ))  
D1 = (IM/WB.WB1) && (IM/WB.dst1 = (ID/IE.src2 && ID/IE.src2 exist ))

A2 = (IM/WB.WB2) && (IM/WB.dst2 = (ID/IE.dst1 && ID/IE.dst1 exist))  
B2 = (IM/WB.WB2) && (IM/WB.dst2 = (ID/IE.dst2 && ID/IE.dst2 exist ))  
C2 = (IM/WB.WB2) && (IM/WB.dst2 = ( ID/IE.src1 && ID/IE.src1 exist ))  
D2 = (IM/WB.WB2) && (IM/WB.dst2 = (ID/IE.src2 && ID/IE.src2 exist ))

## ALU Forward:

AluF = E1 || F1 || G1 || H1 || E2 || F2 || G2 || H2

E1 = (IE/IM.WB1) && (IE/IM.dst1 = (ID/IE.dst1 && ID/IE.dst1 exist))  
F1 = (IE/IM.WB1) && (IE/IM.dst1 = (ID/IE.dst2 && ID/IE.dst2 exist ))  
G1 = (IE/IM.WB1) && (IE/IM.dst1 = ( ID/IE.src1 && ID/IE.src1 exist ))  
H1 = (IE/IM.WB1) && (IE/IM.dst1 = (ID/IE.src2 && ID/IE.src2 exist ))

E2 = (IE/IM.WB2) && (IE/IM.dst2 = (ID/IE.dst1 && ID/IE.dst1 exist))  
F2 = (IE/IM.WB2) && (IE/IM.dst2 = (ID/IE.dst2 && ID/IE.dst2 exist ))  
G2 = (IE/IM.WB2) && (IE/IM.dst2 = ( ID/IE.src1 && ID/IE.src1 exist ))  
H2 = (IE/IM.WB2) && (IE/IM.dst2 = (ID/IE.src2 && ID/IE.src2 exist ))