Some States and their functions (More state are described later in the FSM part):

> **Fetch**:
Fetch instruction from memory (address given in PC) and load in register. Increment PC by 4.
PW=1, IW=1

>**RdAB:**
Load registers A and B
For MUL/MLA instruction, A= Ins[15-12] and B= Ins[3-0], hence **r1src = 01, r2src = 0.**
For other, A=Ins[19-16] and B=Ins[3-0]., hence **r1src=00, r2src=0**.
In both cases, **AW=1, BW=1**

>**RdBC:**
Load registers B and C (Needed for DP instructions where register amount is specified by register (Rs), and MUL/MLA instructions where Rd= Rm*Rs + Rn). If instruction decoder output is for **arith** sub-class of DP, transition to that state (**reg_shift_reg)**, else if it is **mul** sub-class, transition to **RdC** state, else if it is **test** sub-class, transition to **test** state.
r1src=11, r2src=0.

>**RdC:**
Read **Rs** into register C, and transition to **mul** state.

>**WriteRes:**
Write contents of **Res** to register file with destination specified by **Rd.**
Set **M2R='0'** , and if MUL type of instruction, set **Wsrc='0'** (for ins[19-16]) and **'1'** in case of Arith/Test (for ins[15-12]).

DP instruction types:
(For decoding)
********** IR[27-26] = 00 **********
And at least one of IR[7] or IR[4] = 0

Arith-
Rd :- IR[15 -12] , Rn :- IR[19-16] , Rm :- IR[3-0], Rs :- IR[11-8]
Shift is specified by IR[11 - 4]
Add rd,rn,#k here the immediate bit will be 1 (Immediate bit :- 25)
Add rd,rn,rm here immediate bit 0
Add rd,rn,rm,LSL #k IR[4] = 0 , IR[6-5] shift type IR[11-7] shift amount 5 bit unsigned
Add rd,rn,rm,LSL rs IR[4] = 0 , IR[7] = 0 , IR[6-5] = shift type and IR[11-8] = shift type
In case of simple arithmetic instruction with shift specified by register fourth bit is 1 and seventh bit is 0 , when shift specified by immediate fourth bit is 0.
MUL/MLA -
In case of mul / mla IR[7-4] = “1001”
In multiplication Rd and Rn are reversed
Ir[21] = 1 for mla and 0 for mul

Test :-
Rn - second Operand s bit = IR[20] = 1 ;

DP FSM:
In **rdAB** state, if class (DP | DT | MUL) output from instruction decoder is DP, we will check the sub-class (arith | mul | test) and variant s (imm | reg_imm | reg_reg), to decide the next state.
In case of **arith-**
> if **imm**, set **Asrc2=010**, **Asrc1=00, ResW=1.** If 'S' bit is set (as pointed by instruction decoder), set the flags. (**Fset=1**). Transition to **WriteRes** state, where contents of Res will be written to Rd.
> if **reg,** set **Asrc2=000, Asrc1=00** rest same as previous.
>if **reg_shift_constant,** set **Asrc2=101 (to set ALU source as shifter output)**, **shift_amt_src='1'** , **Asrc1=00** and **ResW=1** , and transition to **WriteRes** state.
>If **reg_shift_reg,** go to **RdBC.** After reading registers B and C, in next clock cycle set **Asrc2=101 (to set ALU source as shifter output)**, **shift_amt_src='0' (for reading register shift amount) , Asrc1=00** and **ResW=1** and transition to **WriteRes** state.

In case of mul-
> Go to **RdBC**. After reading registers B and C, in next clock cycle, go to **RdC** state (to read **Rs** into A). In next state, set **Asrc2=110 (to set ALU source as multiplier output)**, and if instruction is **mul** then **Asrc1=010 (for zero addition),** and in case of **mla Asrc2=00** (**for Rs addition)**. Then transition to **WriteRes** state.

Inc case of test-
>Go to **RdBC.** After reading registers B and C, set **Asrc1=00** and **Asrc2=000** and set flags (**Fset=1)**.

DT FSM-

In **rdAB** state, if class (DP | DT | MUL) output from instruction decoder is DT, we will proceed to the following state-

> In first cycle B = Rm , A = Rn .

> If immediate = 1 then set **Asrc2 = “010” ,** otherwise set **Asrc2 = “101”.**

> In case of pre indexing **IorD = “10” else “01”**

> In second cycle keep **A = Rn and B = Rd**

> Then **wad = Rd , M2R = ‘1**’

> If write back is 1 then wad = Rn and M2R = ‘0’

DT INSTRUCTION (For 4 bytes hw/bit later)

********** IR[27-26] = 01 **********

IR[25] = Immediate

IR[24] = Pre/Post Indexing bit 0 -> post

IR[23] = Up/Down bit 0 -> subtract else add

IR[22] = 0 -> transfer word else byte

IR[21] = 0 -> No write back else write back

IR[20] = 0 -> Store else Load

Rn = IR[19-16]

Rd = IR[15 -12]

Offset = IR[11-0]

Condition = IR[31 -28]

When offset is immediate = IR[11-0] otherwise same shifting of register as in DP instruction

Halfword / Signed data transfer

NOTE it will also have IR[27-26] = “00”
IR[11-8] = “0000” and IR[4] = 1 , IR[7] = 1 and IR[6-5] != “00”

IR[24] = Pre/Post Indexing bit 0 -> post
IR[23] = Up/Down bit 0 -> subtract else add
IR[21] = 0 -> No write back else write back
IR[20] = 0 -> Store else Load
Here offset can only be a register
Rm = IR[3-0]
Rn = IR[19-16]
Rd = IR[15-12]

Branch Instruction
IR[27-26] = “10”

IR[24] = Link bit 0 -> branch else branch with link
IR[23-0] = offset

Branch / Branch Link
For branch(no link)
Ascr2 = “011”
Asrc1 = “01” (Input)
PCwrite = 1 ;
In case of Link, will have to write PC to r14. Hence we will have to add another input to multiplexer feeding into wd port of register file. Also, Wad needs to have r14 as well as one of its inputs.