# ECE 7530 ADVANCED DIGITAL VLSI USING VHDL
# Winter'2011 PROJECTS

## Project Grading:

Each group is allowed a maximum of 40 minutes presentation; each member must present his/her contribution to the project. The group grade depends on the completeness of the project, and quality of presentation. Each member is graded individually based on his or her contribution to the project. A group member who fails to present his/her part will receive at-most 50% of the group grade. A group member who did not contribute anything will be given 0%.

**Presentation Dates:**
April 20, 2011 (Wed) 2 Groups

**Report must include the following:**

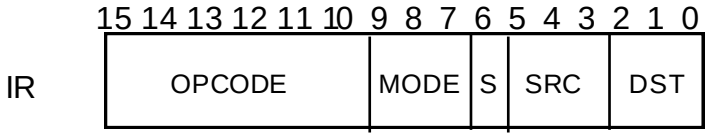
1. The VHDL code for each module of the CISC computer.
2. The VHDL code of the controller, which generates the control signals for each instruction in your project.
3. Test results showing that each instruction is working correctly.
4. The machine-code that implements the program specified in your project. Comment each code by indicating the operation being performed using register transfer notation.
5. Synthesize the VHDL code. Show the resulting schematic diagram.
6. Must provide the list of tasks assign to each member in the group. Member without assignment will not get any credit for the project.

**NOTE:** If your group has successfully completed the project, your group must also be ready to demonstrate it in the Lab for any random operands.

**April 20, 2011 (Wednesday)**

| Presentation Order | Group # | Project # |
|---|---|---|
| 1 | 1 | 1 |
| 2 | 2 | 3 |

INSTRUCTION REGISTER (IR) FORMAT:




OPCODE =Operation Code
MODE= Addressing mode (8)
S=0: MODE apply to SRC --For two-operand case
=1: MODE apply to DST
SRC= Source Address --For two-operand case
DST= Destination Address

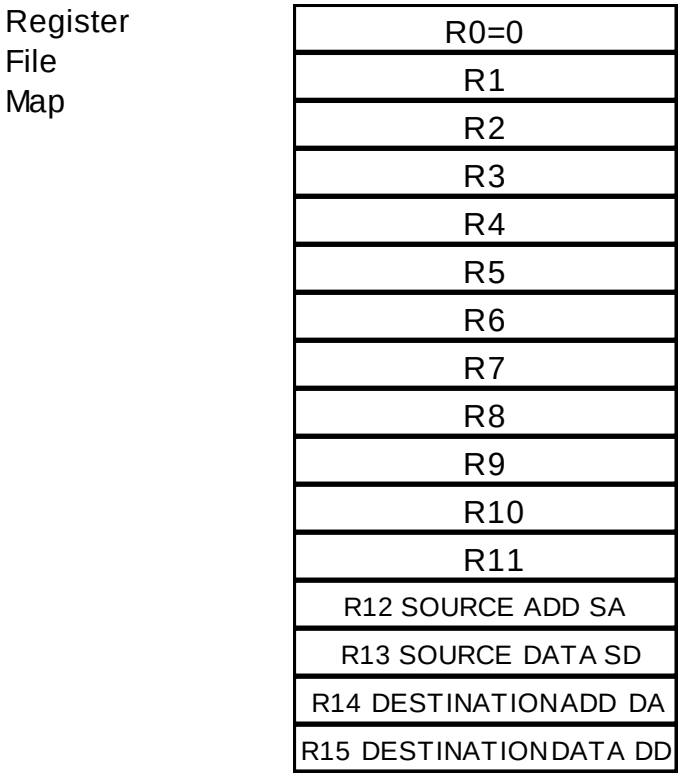



**Figure 1. Register File and Instruction Format**

## 1ST Cycle Instruction Fetch:

IR<=M[PC], PC<=PC+1

TABLE 1. One-Operand Instruction Cycle-by-Cylce Operations

| Instruction | Mode | 2nd Cycle | 3rd Cycle | 4th Cycle | 5th Cycle | 6th Cycle | 7th Cycle |
|---|---|---|---|---|---|---|---|
| **INC R[DST],RG** | **Register (000)** | {data1}<br>DD<=R[DST] | {1EX}<br>DD<=DD+1 | {Writeback}<br>R[DST]<=DD | | | |
| **INC R[DST],RGI** | **Register Indirect (001)** | {addr1}<br>DA<=R[DST] | {data1}<br>DD<=M[DA] | {1EX}<br>DD<=DD+1 | {Writeback}<br>M[DA]<=DD | | |
| **INC W,IM** | **Immediate (010)** | - | - | | | | |
| **INC W,DR** | **Direct (011)** | {addr1}<br>DA<=M[PC]<br>PC<=PC+1 | {data1}<br>DD<=M[DA] | {1EX}<br>DD<=DD+1 | {Writeback}<br>M[DA]<=DD | | |
| **INC W,R[DST],ID** | **Indexed (100)** | {addr1}<br>DA<=M[PC]<br>PC<PC+1 | {addr2}<br>DA<=DA+R[DST] | {data}<br>DD<=M[DA] | {1EX}<br>DD<=DD+1 | {Writeback}<br>M[DA]<=DD | |
| **INC W,R[DST],IDI** | **Indexed Indirect (101)** | {addr1}<br>DA<=M[PC]<br>PC<=PC+1 | {addr2}<br>DA<=DA+R[DST] | {addr3}<br>DA<=M[DA] | {data1}<br>DD<=M[DA] | {1EX}<br>DD<=DD+1 | {Writeback}<br>M[DA]<=DD |
| **INC W,RL** | **Relative (110)** | {addr1}<br>DA<=M[PC]<br>PC<=PC+1 | {addr2}<br>DA<=DA+PC | {data1}<br>DD<=M[DA] | {1EX}<br>DD<=DD+1 | {Writeback}<br>M[DA]<=DD | |
| **INC W,RLI** | **Relative Indirect (111)** | {addr1}<br>DA<=M[PC]<br>PC<=PC+1 | {addr2}<br>DA<=DA+PC | {addr3}<br>DA<=M[DA] | {data1}<br>DD<=M[DA] | {1EX}<br>DD<=DD+1 | {Writeback}<br>M[DA]<=DD |

TABLE 2. Two-Operand Instruction Cycle-by-Cylce Operations

| Instruction | Mode | 2nd Cycle | 3rd Cycle | 4th Cycle | 5th Cycle | 6th Cycle | 7th Cycle | 8th Cycle |
|---|---|---|---|---|---|---|---|---|
| **ADD R[SRC],R[DST]** | **Register S=0** | {data1} SD<=R[SRC] | {data2} DD<=R[DST] | {2EX} DD<=DD+SD | {Writeback} R[DST]<=DD | | | |
| **ADD R[DST],R[SRC]** | **Register S=1** | {data1} DD<=R[DST] | {data2} SD<=R[SRC] | {2EX} DD<=DD+SD | {Writeback} R[DST]<=DD | | | |
| **ADD R[SRC]I,R[DST]** | **Register Indirect S=0** | {addr1} SA<=R[SRC] | {data1} SD<=M[SA] | {data2} DD<=R[DST] | {2EX} DD<=DD+SD | {Writeback} R[DST]<=DD | | |
| **ADD R[DST]I,R[SRC]** | **Register Indirect S=1** | {addr1} DA<=R[DST] | {data1} DD<=M[DA] | {data2} SD<=R[SRC] | {2EX} DD<=DD+SD | {Writeback} M[DA]<=DD | | |
| **ADD W,IM** | **Immediate S=0** | {data1} SD<=M[PC] PC<=PC+1 | {data2} DD<=R[DST] | {2EX} DD<=DD+SD | {WrItebacck} R[DST]<=DD | | | |
| **ADD W,IM** | **Immediate S=1** | {data1} DD<=M[PC] PC<=PC+1 | {data2} SD<=R[SRC] | {2EX} DD<=DD+SD | {WrItebacck} R[DST]<=DD | | | |
| **ADD W,DR** | **Direct S=0** | {addr1} SA<=M[PC] PC<=PC+1 | {data1} SD<=M[SA] | {data2} DD<=R[DST] | {2EX} DD<=DD+SD | {Writeback} R[DST]<=DD | | |
| **ADD W,DR** | **Direct S=1** | {addr1} DA<=M[PC] PC<=PC+1 | {data1} DD<=M[DA] | {data2} SD<=R[SRC] | {2EX} DD<=DD+SD | {Writeback} M[DA]<=DD | | |
| **ADD W,R[SRC]I,R[DST]** | **Indexed S=0** | {addr1} SA<=M[PC] PC<PC+1 | {addr2} SA<=SA+R[SRC] | {data1} SD<=M[SA] | {data2} DD<=R[DST] | {2EX} DD<=DD+SD | {Writeback} R[DST]<=DD | |
| **ADD W,R[SRC]I,R[DST]** | **Indexed S=1** | {addr1} DA<=M[PC] PC<PC+1 | {addr2} DA<=DA+R[DST] | {data1} DD<=M[DA] | {data2} SD<=R[SRC] | {2EX} DD<=DD+SD | {Writeback} M[DA]<=DD | |
| **ADD W,R[SRC]II,R[DST]** | **Indexed Indirect S=0** | {addr1} SA<=M[PC] PC<=PC+1 | {addr2} SA<=SA+R[SRC] | {addr3} SA<=M[SA] | {data1} SD<=M[SA] | {data2} DD<=R[DST] | {2EX} DD<=DD+SD | {Writeback} R[DST]<=DD |
| **ADD W,R[SRC]II,R[DST]** | **Indexed Indirect S=1** | {addr1} DA<=M[PC] PC<=PC+1 | {addr2} DA<=DA+R[DST] | {addr3} DA<=M[DA] | {data1} DD<=M[DA] | {data2} SD<=R[SRC] | {2EX} DD<=DD+SD | {Writeback} M[DA]<=DD |
| **ADD W,RL** | **Relative S=0** | {addr1} SA<=M[PC] PC<=PC+1 | {addr2} SA<=SA+PC | {data1} SD<=M[SA] | {data2} DD<=R[DST] | {2EX} DD<=DD+SD | {Writeback} R[DST]<=DD | |
| **ADD W,RL** | **Relative S=1** | {addr1} DA<=M[PC] PC<=PC+1 | {addr2} DA<=DA+PC | {data1} DD<=M[DA] | {data2} SD<=R[SRC] | {2EX} DD<=DD+SD | {Writeback} M[DA]<=DD | |
| **ADD W,RLI** | **Relative Indirect S=0** | {addr1} SA<=M[PC] PC<=PC+1 | {addr2} SA<=SA+PC | {addr3} SA<=M[SA] | {data1} SD<=M[SA] | {data2} DD<=R[DST] | {1EX} DD<=DD+SD | {Writeback} R[DST]<=DD |
| **ADD W,RLI** | **Relative Indirect S=1** | {addr1} DA<=M[PC] PC<=PC+1 | {addr2} DA<=DA+PC | {addr3} DA<=M[DA] | {data1} DD<=M[DA] | {data2} SD<=R[SRC] | {1EX} DD<=DD+SD | {Writeback} M[DA]<=DD |

**For Two Operands Instructions: S=0 => MODE Apply to SRC**
**S=1=> MODE Apply to DST**

Branch address Fetch is similar to One-operand Fetch, with some exceptions: We are interested in destination address rather than destination operand, i.e. branch execution will start as soon as the destination address has been placed in DA. As a result, the register mode (Mode(2:0)="000") is invalid.

## Branch instruction execution

Based on the OPCODE(3:0), there is a 16-way branch, we can categorize as three types: unconditional jump("0000"), call procedure("0001"), the rest are those conditional branches based on the PSR bit.

TABLE 3. The BEX cycle for each branch instruction:

| Opcode(5..0) | Instruction | 2nd Cycle | 3rd Cycle | 4th Cycle | 5th Cycle |
|---|---|---|---|---|---|
| **110000** | **JMP** | PC <= DA | | | |
| **110001** | **CALL** | R8<=PC | SP<=SP-1 | M[SP]<=R8 | PC<=DA |
| **111000** | **BZ** | PC <= DA (if Z ='1') | | | |
| **111001** | **BNZ** | PC <= DA (if Z='0') | | | |
| **111010** | **BC** | PC <= DA (if C ='1') | | | |
| **111011** | **BNC** | PC <= DA (if C ='0') | | | |
| **111100** | **BN** | PC <= DA (if N='1') | | | |
| **111101** | **BNN** | PC <= DA (if N ='0') | | | |
| **111110** | **BV** | PC <= DA (if V='1') | | | |
| **111111** | **BNV** | PC <= DA (if V='0') | | | |

## Datapath Control Signals:

**DSA[4..0]** Destination & Source Address, 5-bit
DSA4 =0 : Register File Address in DSA[3..0]
=1* : Register File Address in the instruction;
0||DST if DSA3=0
0||SRC if DSA3=1

**SB** Source B address, 5-bit
SB4 =0 : Register File Address in SB[3..0]
=1* : Register File Address in the instruction;
0||DST if SB3=0
0||SRC if SB3=1

* Only one can be 1, either DSA4=1 or SB4=1 but not both.

**MA** MUXA select signal, 2-bit
0 A data
1 PC
2 SP
3 B data

**MB** MUXB select signals, 1-bit
0 B data
1 0…0&PSR (11zeroes+5 bit PSR)

**MD** MUXD select signal, 1-bit
0 Function
1 M[A data]

**FS** ALU function select signals, 6-bit

**TABLE 4. Miscellaneous Operation (MO*) Control Signal**

| **Control Signal** | **Mnemonic** |
|---|---|
| Increment PC | IncPC |
| Load PC | LdPC |
| Increment SP | IncSP |
| Decrement SP | DecSP |
| Load SP | LdSP |
| Load Instruction register | LdIR |
| Load status bits Z,C,N,V | LdZCNV |
| Load status bits Z, C | LdZC |
| Load status bits Z, N | LdZN |
| Register write | RW |
| Memory write | MW |

**TABLE 5. Processor Status Register (PSR) Control Signal Truth Table:**

| Input | | | Output | | | |
|---|---|---|---|---|---|---|
| **LdZCNV** | **LdZC** | **LdZN** | **ZLD** | **CLD** | **NLD** | **VLD** |
| 1 | 0 | 0 | 1 | 1 | 1 | 1 |
| 0 | 1 | 0 | 1 | 1 | 0 | 0 |
| 0 | 0 | 1 | 1 | 0 | 1 | 0 |

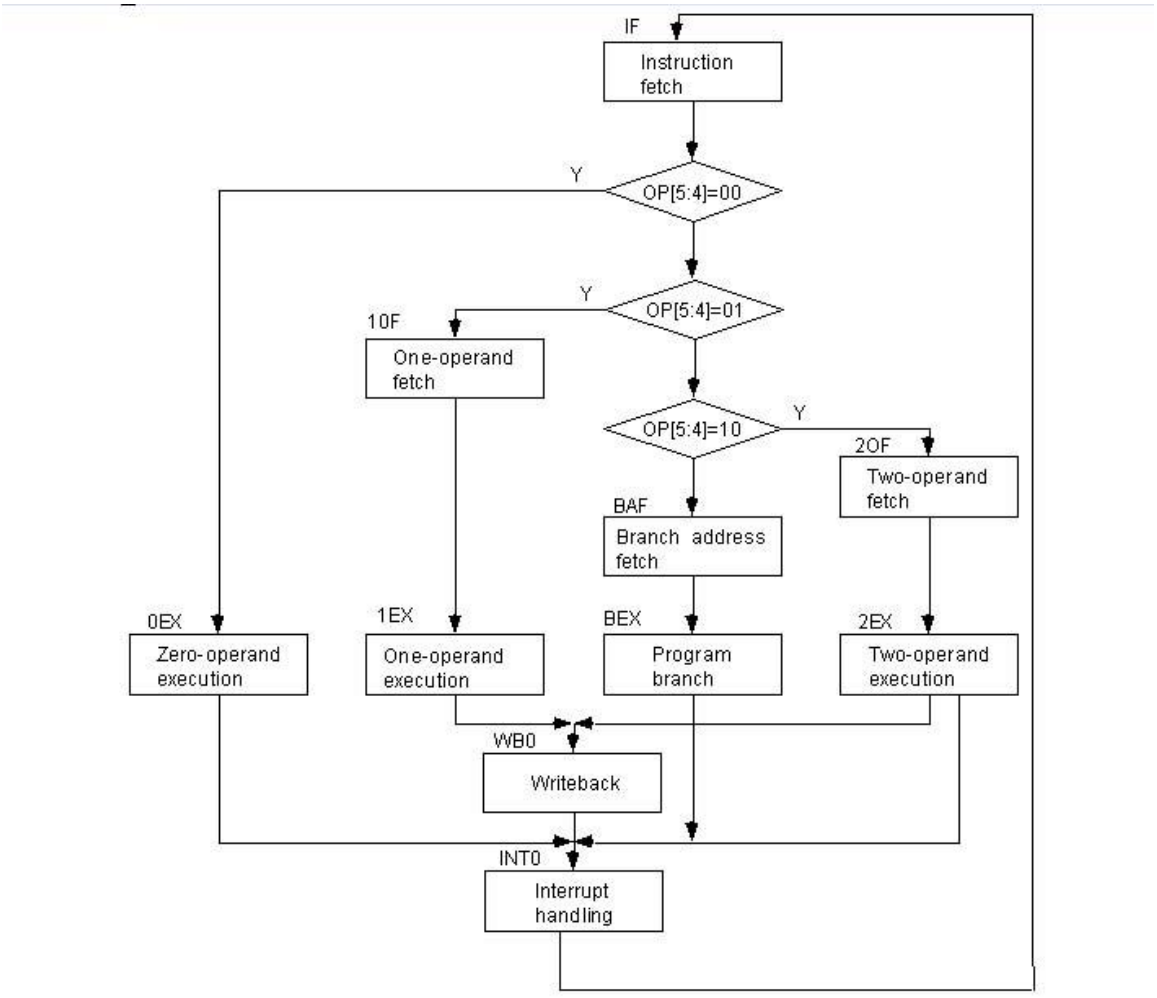
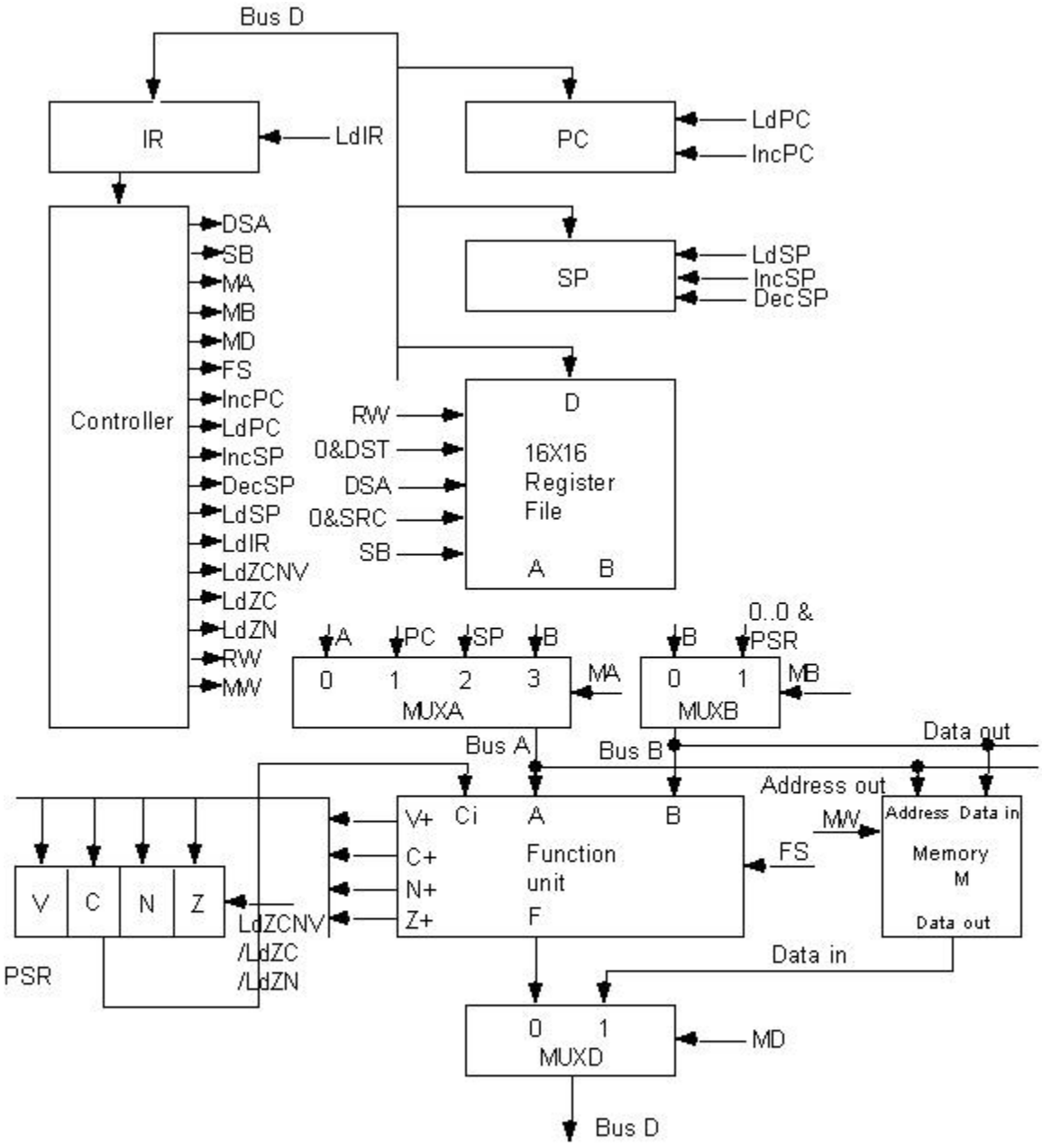



Figure 2. CPU Operation Flowchart.




**CISC CPU**

Figure 3. CPU Block Diagram

## 1ST Cycle Instruction Fetch:

IR<=M[PC], PC<=PC+1

### TABLE 6. The instructions to be used in the project are listed below:

| Instruction | Mode | 2nd C ycle | 3rd Cycle | 4th Cycle | 5th Cycle | 6th Cycle | 7th Cycle |
|---|---|---|---|---|---|---|---|
| **INC R[DST],RG** | **Register (000)** | {data1} DD<=R[DST] | {1EX} DD<=DD+1 | {Writeback} R[DST]<=DD | | | |
| **DEC R[DST],RG** | **Register (000)** | {data1} DD<=R[DST] | {1EX} DD<=DD-1 | {Writeback} R[DST]<=DD | | | |
| **ADD R[SRC],R[DST]** | **Register S=0** | {data1} SD<=R[SRC] | {data2} DD<=R[DST] | {2EX} DD<=DD+SD | {Writeback} R[DST]<=DD | | |
| **SUB R[SRC],R[DST]** | **Register S=0** | {data1} SD<=R[SRC] | {data2} DD<=R[DST] | {2EX} DD<=DD-SD | {Writeback} R[DST]<=DD | | |
| **RORC R[DST],RG** | **Register (000)** | {data1} DD<=R[DST] | {1EX} DD<=Cin & DD(15:1) | {Writeback} R[DST]<=DD | | | |
| **BNC W** | **Immediate (010)** | DA<=M[PC] PC<PC+1 | {BEX} PC<=DA If C≠'1' | | | | |
| **BNZ W** | **Immediate (010)** | DA<=M[PC] PC<PC+1 | {BEX} PC<=DA If Z≠'1' | | | | |
| **BNN W** | **Immediate (010)** | DA<=M[PC] PC<PC+1 | {BEX} PC<=DA If N≠'1' | | | | |
| **BNC M[W]** | **Direct (011) S=0** | {addr1} DA<=M[PC] PC<PC+1 | {addr2} DA<=M[DA] | {BEX} PC<=DA If C≠'1' | | | |
| **BNZ M[W]** | **Direct (011) S=0** | {addr1} DA<=M[PC] PC<PC+1 | {addr2} DA<=M[DA] | {BEX} PC<=DA If Z≠'1' | | | |
| **BNN M[W]** | **Direct (011) S=0** | {addr1} DA<=M[PC] PC<PC+1 | {addr2} DA<=M[DA] | {BEX} PC<=DA If N≠'1' | | | |
| **MOVE [SRC], [DST]** | **Register (000)** | {data 1} SD<=R[SRC] | {data 2} DD<=R[DST] | {2EX} DD<=SD | {Writeback} R[DST]<=DD | | |
| **MOVE W,DR** | **Direct (011) S=0** | {addr1} SA<=M[PC] PC<=PC+1 | {data1} SD<=M[SA] | {data2} DD<=R[DST] | {2EX} DD<=SD | {Writeback} R[DST]<=DD | |
| **MOVE W,DR** | **Direct (011) S=1** | {addr1} DA<=M[PC] PC<=PC+1 | {data1} DD<=M[DA] | {data2} SD<=R[SRC] | {2EX} DD<=SD | {Writeback} M[DA]<=DD | |

**TABLE 7. Instruction & Data Fetch Control Signals:**

| | DSA[5] | SB[5] | MA[2] | MB[1] | MD[1] | FS[6] | MO* |
|---|---|---|---|---|---|---|---|
| **{Inst Fetch}** | | | | | | | |
| IR<=M[PC]<br>PC<=PC+1 | 00 | 00 | 1(PC) | 0 | 1(M[PC]) | 00 | LdIR<br>IncPC |
| **{data1}** | | | | | | | |
| DD<=R[DST] | 0F(DD) | 10(R[DST]) | 0(A) | 0(B) | 0(F) | 20(B) | RW |
| DD<=M[DA] | 0F(DD) | 0E(DA) | 3(B) | 0 | 1(M[DA]) | 00 | RW |
| **SD<=R[SRC]** | **0D(SD)** | **18(R[SRC])** | **0(A)** | **0(B)** | **0(F)** | **20(B)** | **RW** |
| **SD<=M[SA]** | **0D(SD)** | **0C(SA)** | **3(B)** | **0** | **1(M[DA])** | **00** | **RW** |
| **SD<=M[PC]<br>PC<=PC+1** | **0D(SD)** | **00** | **1(PC)** | **0** | **1(M[PC])** | **00** | **RW<br>IncPC** |
| **DD<=M[PC]<br>PC<=PC+1** | **0F(DD)** | **00** | **1(PC)** | **0** | **1(M[PC])** | **00** | **RW<br>IncPC** |
| **{data2}** | | | | | | | |
| **DD<=R[DST]** | **0F(DD)** | **10(R[DST])** | **0** | **0** | **0** | **20(B)** | **RW** |
| **SD<=R[SRC]** | **0D(SD)** | **18(R[SRC])** | **0** | **0** | **0** | **20(B)** | **RW** |
| **{addr1}** | | | | | | | |
| DA<=R[DST] | 0E(DA) | 10(R[DST]) | 0 | 0 | 0 | 20(B) | RW |
| DA<=M[PC];<br>PC<=PC+1 | 0E(DA) | 00 | 1(PC) | 0 | 1(M[PC]) | 00 | IncPC<br>RW |
| **SA<=M[PC];<br>PC<=PC+1** | **0C(SA)** | **00** | **1(PC)** | **0** | **1(M[PC])** | **00** | **IncPC<br>RW** |
| **SA<=R[SRC]** | **0C(SA)** | **18(R[SRC])** | **0** | **0** | **0** | **20(B)** | **RW** |
| **DA<=R[DST]** | **0E(DA)** | **10(R[DST])** | **0** | **0** | **0** | **20(B)** | **RW** |
| **{addr2}** | | | | | | | |
| DA<=DA+R[DST] | 0E(DA) | 10(R[DST]) | 0 | 0 | 0 | 04(+) | RW |
| DA<=DA+PC | 0E(DA) | 0E(DA) | 1(PC) | 1 | 0 | 04(+) | RW |
| **SA<=R[SRC]+SA** | **0C(SA)** | **18(R[SRC])** | **0** | **0** | **0** | **04(+)** | **RW** |
| **SA<=SA+PC** | **0C(SA)** | **0C(SA)** | **1(PC)** | **1** | **0** | **04(+)** | **RW** |
| **{addr3}** | | | | | | | |
| DA<=M[DA] | 0E(DA) | 00 | 0(A) | 0 | 1(M[DA]) | 00 | RW |
| **SA<=M[SA]** | **0C(SA)** | **00** | **0(A)** | **0** | **1(M[SA])** | **00** | **RW** |
| **{Writeback}**** | | | | | | | |
| R[DST]<=DD | 10(R[DST]) | 0F(DD) | 0(A) | 0(B) | 0(F) | 20(B) | RW |
| M[DA]<=DD | 0E(DA) | 0F(DD) | 0(A) | 0(B) | 0(X) | 00(X) | MW |

*Any MO control signal not mention means its value is zero.

** To determine where to put the contents of DD, it is necessary to examine S & MODE

**TABLE 8. Writeback Operation:**

| S | MODE | Writbeback |
|---|---|---|
| 0 | X | R[DST]<=DD |
| 1 | 000 | R[DST]<=DD |
| 1 | ≠000 | M[DA]<=DD |

**TABLE 9. Control signal for subtraction , decrement by 1, Rotate right with carry**

| | DSA[5] | SB[5] | MA[2] | MB[1] | MD[1] | FS[6] | MO* |
|---|---|---|---|---|---|---|---|
| **{SUB}{2EX}** | | | | | | | |
| DD<=DD-SD | 0F(DD) | 0D(SD) | 0(A) | 0(B) | 0(F) | 09(SUB) | LdZCNV RW |
| **{DEC}{1EX}** | | | | | | | |
| DD<=DD-1 | 0F(DD) | 00 | 0(A) | 0(B) | 0(F) | 0C(DEC) | LdZCNV RW |
| {**RORC**} | | | | | | | |
| DD<=Cin & DD(15:1) | 0F(DD) | 0F(DD) | 3(B) | 0(B) | 0(F) | 27(RORC) | LdZC RW |
| **{BEX}** | | | | | | | |
| PC<=DA | 0E(DA) | 00 | 0(A) | 0(B) | 0(F) | 00(A) | LdPC |

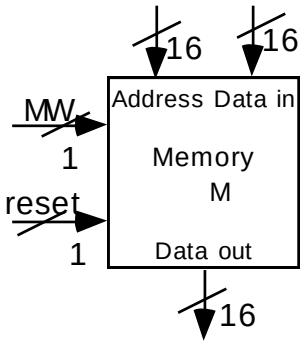
**TABLE 10. Set of Instruction and addressing mode for each project**

| **Multiplication** | | **Division** | |
|---|---|---|---|
| **Project #1** | **Project #2** | **Project #3** | **Project #4** |
| BNC(Immediate) | BNC(Direct) | BNN(Immediate) | BNN(Direct) |
| BNZ(Direct) | BNZ(Immediate) | SUB(Register) | |
| ADD(Register) | | | |
| INC(Register) | | DEC(Register) | |
| RORC(Register) | | INC(Register) | |
| Move(Direct)/Move(Register) | | | |

**TABLE 11. Instruction Opcode and Status Flag Affected for the project**

| **Instruction** | **Mnemonic** | **Mode** | **Opcode** | **Status effect** | **FS** |
|---|---|---|---|---|---|
| Addition | ADD | 000(Reg) | 100010 | ZCNV | 04 |
| Subtraction | SUB | 000(Reg) | 100100 | ZCNV | 09 |
| Increment | INC | 000(Reg) | 010010 | ZCNV | 01 |
| Decrement | DEC | 000(Reg) | 010011 | ZCNV | 0C |
| Rotate right with carry | RORC | 000(Reg) | 011110 | ZC | 27 |
| Branch if no carry | BNC | 010(Imm) | 111011 | None | -- |
| Branch if no carry | BNC | 011(Dir) | 111011 | None | -- |
| Branch if not | BNZ | 010(Imm) | 111001 | None | -- |

| | | | | | |
|---|---|---|---|---|---|
| zero | | | | | |
| Branch if not zero | BNZ | 011(Dir) | 111001 | None | -- |
| Branch if not negative | BNN | 010(Imm) | 111101 | None | -- |
| Branch if not negative | BNN | 011(Dir) | 111101 | None | -- |
| Move register to register | MOVE | 000(Reg) | 100000 | None | -- |
| Move mem to reg | MOVE | 011(dir) | 100000 | None | -- |
| MOVE reg to mem | MOVE | 011(dir) | 100000 | None | -- |




TASK:

1. Write a VHDL model of a memory unit that has:

16-bit Address, Data in, and Data out

MW : 1-bit memory write control signal: 0=read; 1=write

Reset: 1-bit reset signal (same as CPU reset signal), a short duration pulse.

**On the rising edge of reset**, the memory unit will be loaded with your program starting in memory location $0000, and data starting at $0020

**On the rising edge of reset signal,** the PC <=0, and the CPU controller is put into FETCH.

For example to enter the memory map in TABLE 12.

M(0)<=X”8182” --R2<=M[20], MOVE,DIR

M(1)<=X”0020” --Operand Address $0020
Etc.

M(32) <= N(numerator)/A(multiplicand) --$0020 =32
M(33) <= D(Denominator)/B(multiplier) --$0021=33
M(34) <= Q(quotient)/P(product MSW) --$0022=34
M(35) <= R(remainder)/P(product LSW) --$0023=35

**Project 1 & 2:**

Write a program that will multiply the operands stored in memory location 0020(Hex) and 0021(Hex) and store the product in location 0022(Hex) and 0023(Hex). The program must be written with the available instructions as shown in TABLE 10 for the given project # only. The machine-code that implement the program specified in your project must be commented by indicating the operation being performed using register transfer notation.

**Project 3 & 4:**

Write a program that will divide the numerator stored in memory location 0020(Hex) and denominator stored in 0021(Hex) and stores the quotient in location 0022(Hex) and the remainder in 0023(Hex). The program must be written with the available instructions as shown in TABLE 10 for the given project # only. The machine-code that implement the program specified in your project must be commented by indicating the operation being performed using register transfer notation.

**TABLE 12. Memory Map: Sample program code**

| Mem Loc (Dec) | Machine Code | Inst | Operation |
|---|---|---|---|
| 0 | 8182 | MOVE, Dir | R2<=M[20] |
| 1 | 0020 | | |
| 2 | 8183 | MOVE, Dir | R3<=M[21] |
| 3 | 0021 | | |
| .. | | | |
| 30 | | | N/A |
| 31 | | | D/B |
| 32 | | | Q/P(MSB) |
| 33 | | | R/P(LSB) |

| **Inst** | **Operation** | **Machine Code** | |
|---|---|---|---|
| MOVE, Dir | R2<=M[20] | 8182<br>=1000\|0001\|1000\|0010 | OP=100000=MOVE<br>MODE=011=DIR<br>S=0 mode apply to SRC<br>DST=010 =REG(2) |
| | | 0020 | SRC Add in MEM=0020 |
| MOVE, Dir | M[23]<=R4 | 81E0<br>=1000\|0001\|1110\|0000 | OP=100000=MOVE<br>MODE=011=DIR<br>S=1 mode apply to DST<br>SRC=100=REG(4) |
| | | 0023 | DST Add in MEM=0023 |
| MOVE, Reg | R4<=R2 | 8014<br>=1000\|0000\|0001\|0100 | OP=100000=MOVE<br>MODE=000=REG<br>S=0 mode apply to SRC<br>SRC=2<br>DST=4 |