CO224 Computer Architecture

Lab 5 - Building a Simple Processor
Part 5–Extended ISA

Group No: 09
E/20/148 Hewawasam A.K.L.
E/20/157 Janakantha S.M.B.G.

Instruction Encoding

The implemented 8-bit single-cycle processor supports an instruction set architecture of 14 instructions. The instructions , assigned opcodes and their functions as follows.

Instruction	Opcode	Function
`loadi`	`0000_0000`	Loads an immediate value to a register
`mov`	`0000_0001`	Move a value in one register to another
`add`	`0000_0010`	Two register data sets are added and write to another register
`sub`	`0000_0011`	Two register data sets are subtracted and write to another register
`and`	`0000_0100`	Do AND operation for two register data sets and write to another register
`or`	`0000_0101`	Do OR operation for two register data sets and write to another register
`j`	`0000_0110`	Jump to given amount of instructions forward or backward
`beq`	`0000_0111`	If two register values are equal then branch to given instruction.
`mult`	`0000_1000`	Two register data sets are multiplied and write to another register
`sll`	`0000_1001`	Do the logical shift left for a value in one register and write to another register

srl	0000_1010	Do the logical shift right for a value in one register and write to another register
sra	0000_1011	Do the arithmetic shift right for a value in one register and write to another register
ror	0000_1100	Do the rotate right operation for a value in one register and write to another register
bne	0000_1101	If two register values are not equal then branch to given instruction.

Instruction Encoding Format

All the 14 instructions are 32 bits and are encoded in the following manner.

OP-CODE (bits 31-24)	RD / OFFSET (bits 23-16)	RT (bits 15-8)	RS / IMM (bits7-0)

Bits(31-24): OP-CODE field identifies the instruction's operation.
Bits(23-16): A register (RD) to be written to in the register file, or an immediate value (OFFSET) for the jump,beq and bne operations.
Bits(15-8) : A register (RT) to be read from in the register file.
Bits(7-0) : A register (RS) to be read from the register file, or an immediate value (IMM) for the loadi,jump,sll,srl,sra, and ror operations.

Examples :

```
loadi 1 0x01    - load the immediate value 0x01 to register 1. (Ignore bits 15-8)
mov 4 1         - copy the value in register 1 to register 4. (Ignore bits 15-8)
add 4 1 2       - add value in register 2 to value in register 1, place the result in register 4
sub 4 1 2       - subtract value in register 2 from the value in register 1, place the result in register 4
mult 6 4 3      - multiply value in register 4 and 3 , place the result in register 6
and 4 1 2       - perform bit-wise AND on values in registers 1 and 2, place the result in register 4
or 4 1 2        - perform bit-wise OR on values in registers 1 and 2, place the result in register 4
j 0x02          - jump 2 instructions forward from the next instruction to be executed
                  (Ignore bits 15-0)
beq 0xFE 1 2    - if values in registers 1 and 2 are equal, branch 2 instructions backward by
                  manipulating the Program Counter)
```

`bne 0x02 1 2` - if values in registers 1 and 2 are not equal, branch 2 instructions forward by manipulating the Program Counter)

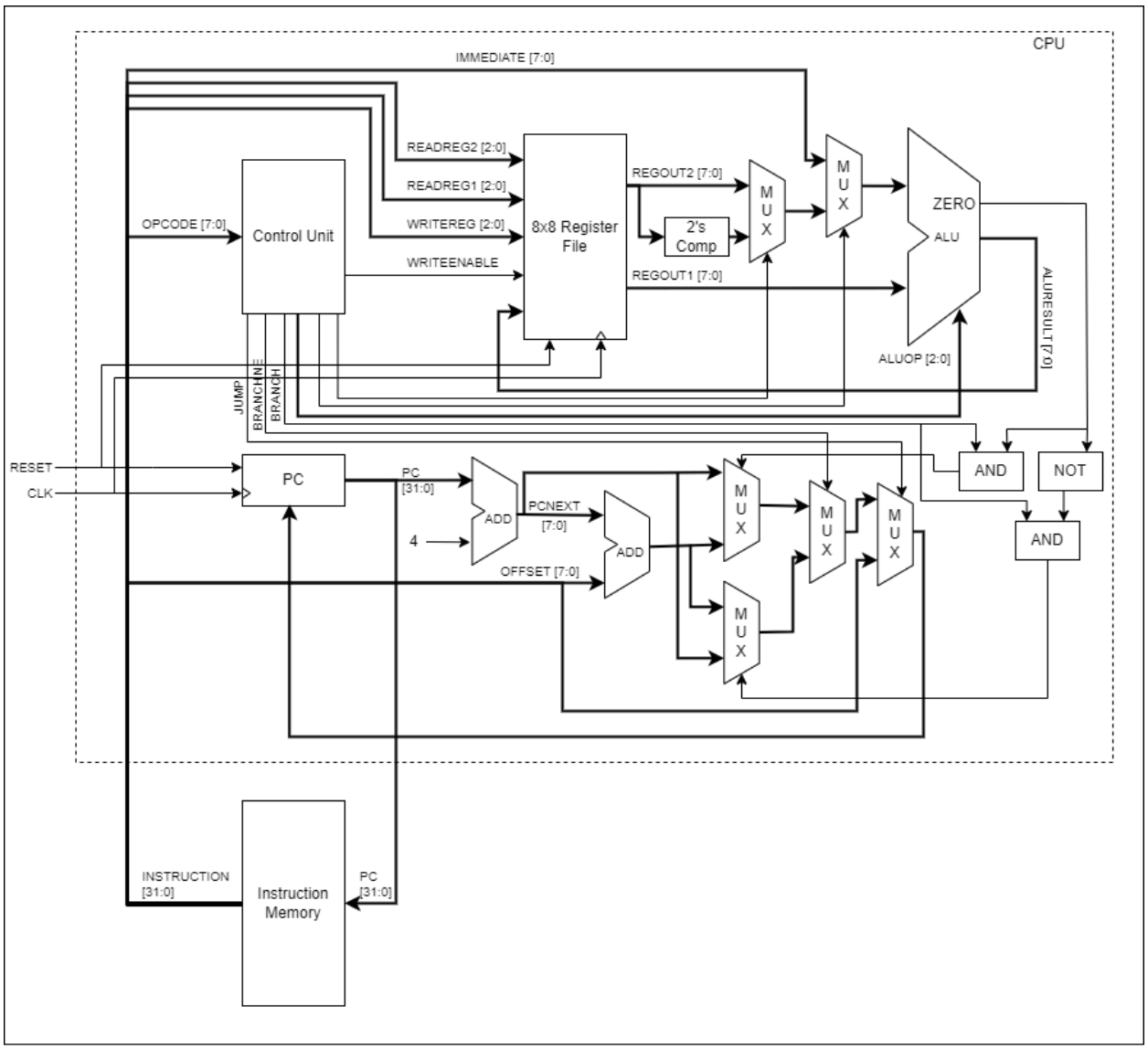
`sll 3 1 0x03` - apply logical shift left 3 times on the value in register 1, place the result in register 3

`srl 4 2 0x04` - apply logical shift right 4 times on the value in register 2, place the result in register 4

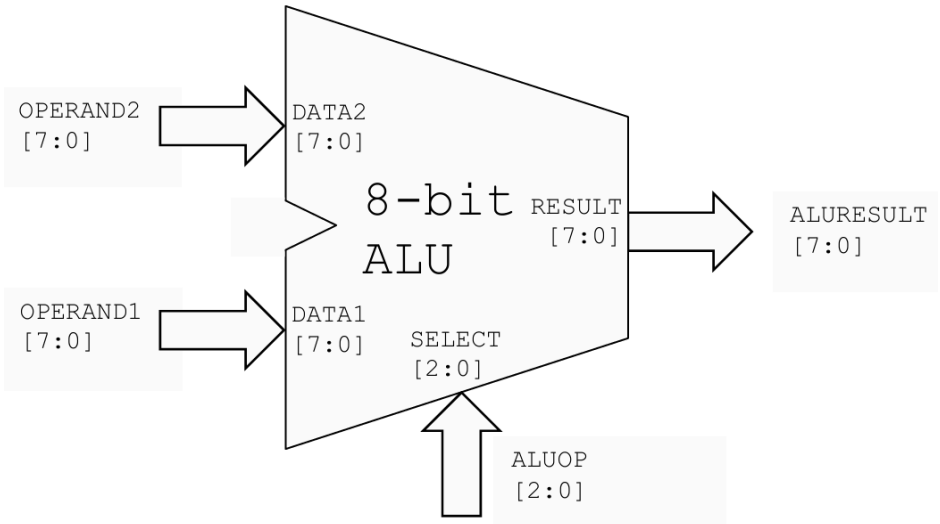
`sra 5 2 0x02` - apply arithmetic shift left 2 times on the value in register 2, place the result in register 5

`ror 6 2 0x03` - apply rotate right 2 times on the value in register 2, place the result in register 6

Overview of CPU



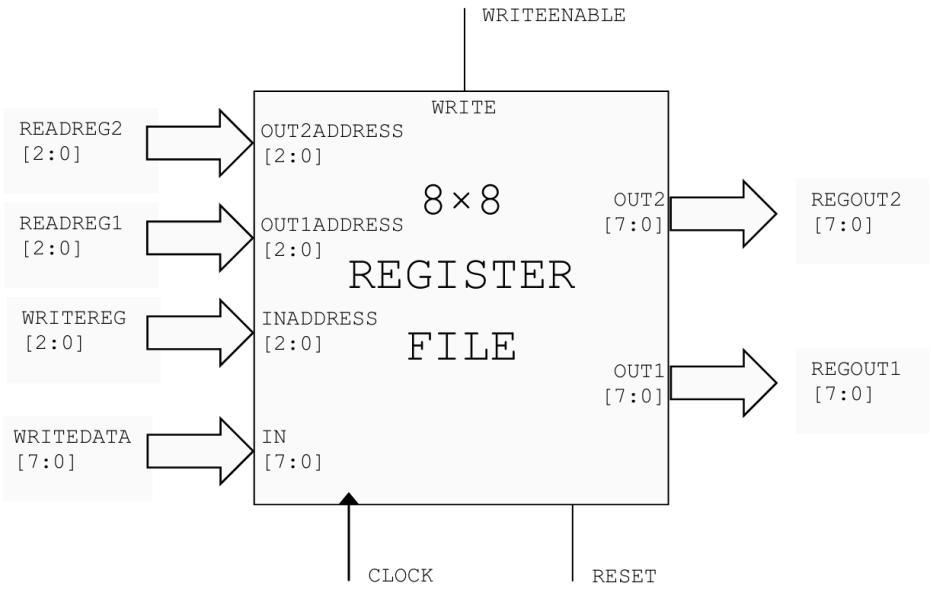
ALU : Arithmetic Logic Unit



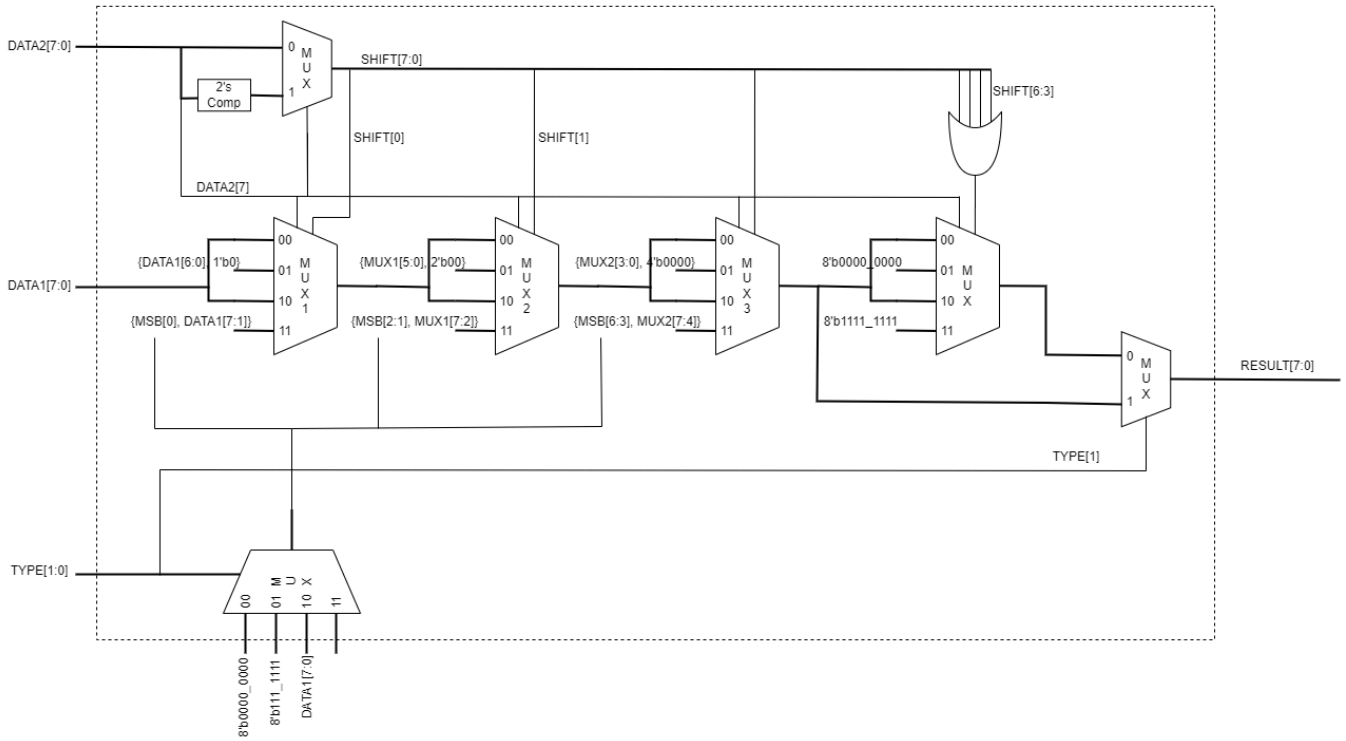
ALU Functions

Function	ALUOP	Supported Instructions
Forward	000	loadi, mov
Add	001	add ,sub
Bitwise And	010	and
Bitwise Or	011	or
Logical Shift	100	sll ,srl
Arithmetic Shift	101	sra
Rotate	110	ror
Multiplication	111	mult

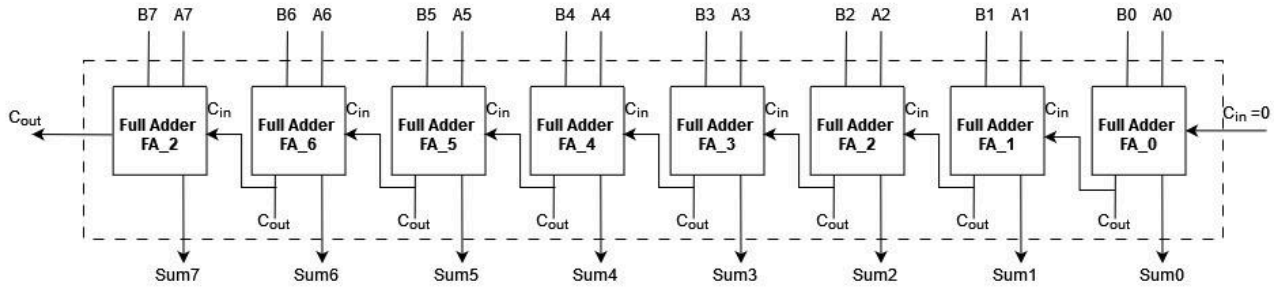
8x8 Register File



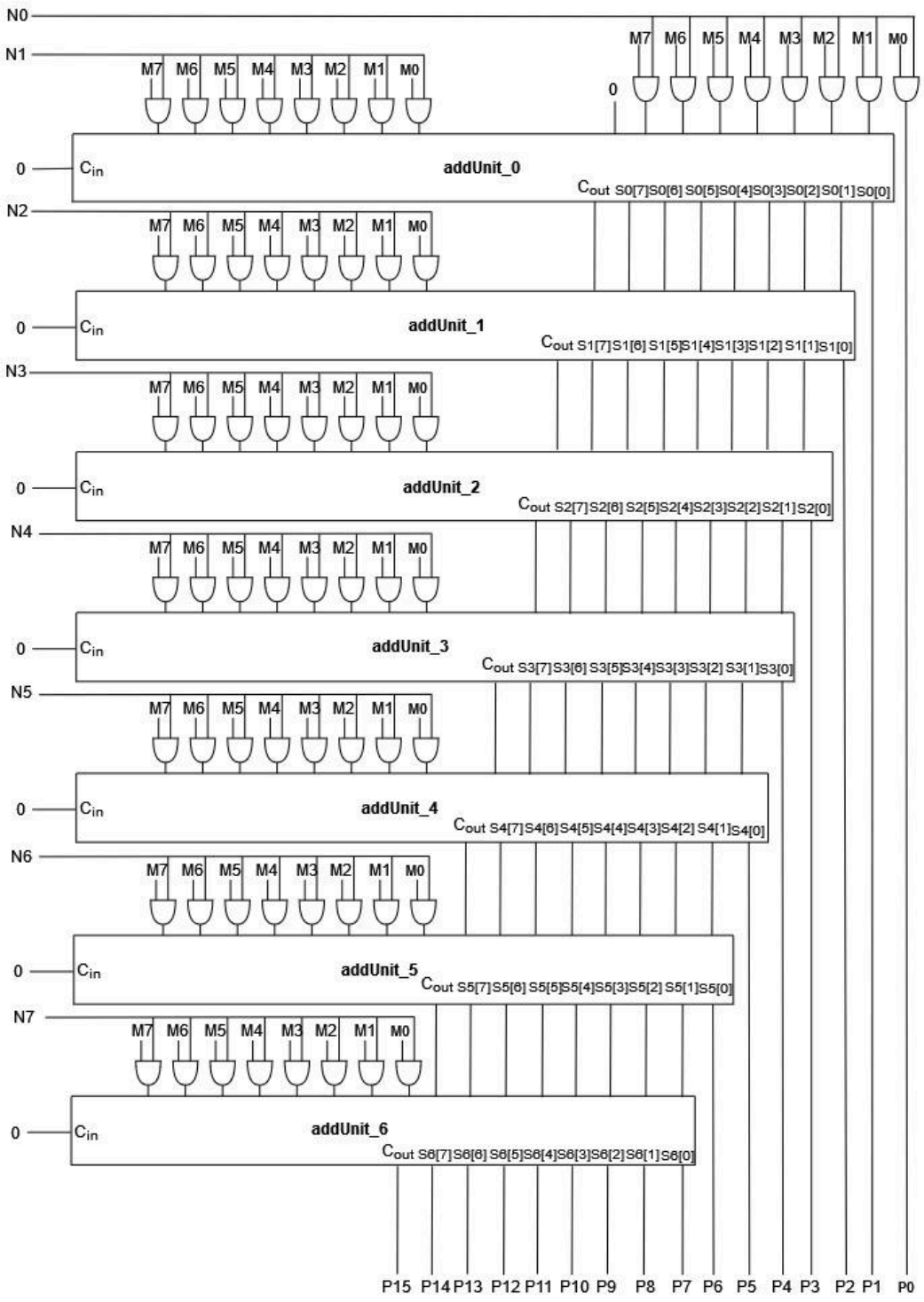
Shift Module



8 bit Adder Module



Multiplication Module



8 bit Multiplicand (M) x 8 bit Multiplier (N) = 16 bit Product (P)

Timing

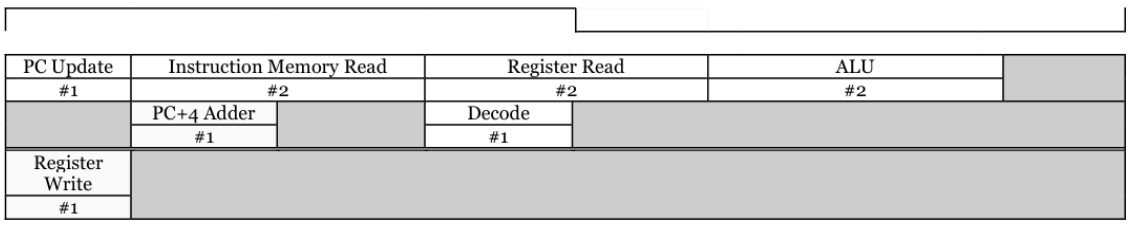
The timing is used to model the real behavior of the actual CPU.

1. Clock Cycle Period : 8 time units

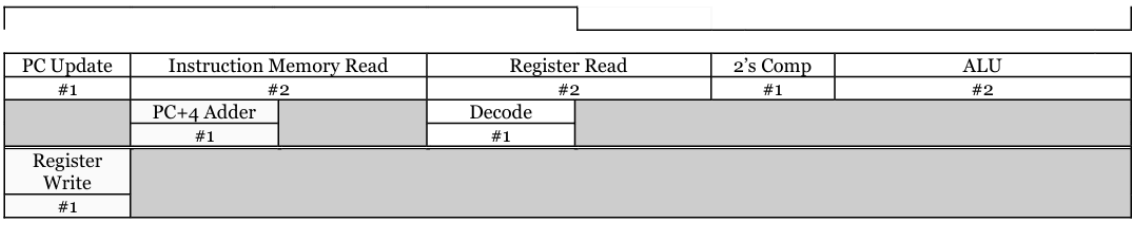
2. Artificial Delays
   - and module : 1 time unit
   - or module : 1 time unit
   - add module : 2 time units
   - 2's complememt module : 1 time unit
   - shift module : 2 time units
   - mul module : 2 time units
   - forward module : 1 time unit
   - Multiplexer modules : No artificial delays

Here are the timing diagrams for main instructions.

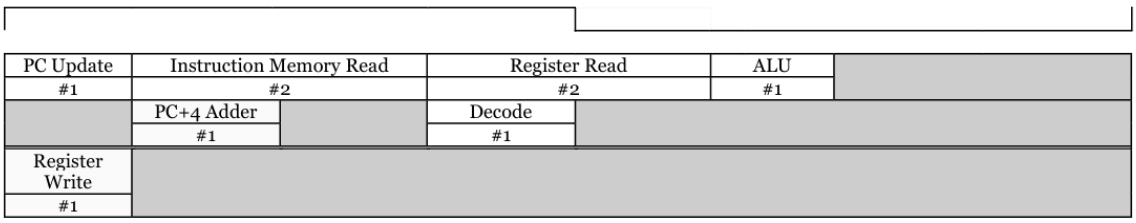
`add:`



`sub:`



`and/or/mov:`



loadi:

PC Update	Instruction Memory Read		ALU	
#1	#2		#1	
	PC+4 Adder		Decode	
	#1		#1	
Register Write				
#1				

j:

PC Update	Instruction Memory Read	Decode	
#1	#2	#1	
	PC+4 Adder	Branch/Jump Target Adder	
	#1	#2	

beq:

PC Update	Instruction Memory Read		Register Read	2's Comp	ALU
#1	#2		#2	#1	#2
	PC+4 Adder		Branch/Jump Target Adder		
	#1		#2		
			Decode		
			#1		

sll/srl/sra/ror:

PC Update	Instruction Memory Read		Register Read		Shifting	
#1	#2		#2		#3	
	PC + 4 Adder		Decode			
	#1		#1			
			Find shifting direction		Find shifting amount	
			#1		#1	
Register Write						
#1						

mult:

PC Update	Instruction Memory Read		Register Read	ALU	
#1	#2		#2	#2	
	PC + 4 Adder		Decode		
	#1		#1		
Register Write					
#1					

bne:

PC Update	Instruction Memory Read		Register Read	2's Comp	ALU
#1	#2		#2	#1	#2
	PC+4 Adder		Branch/Jump Target Adder		
	#1		#2		
			Decode		
			#1		