11 - register transfer and microoperations

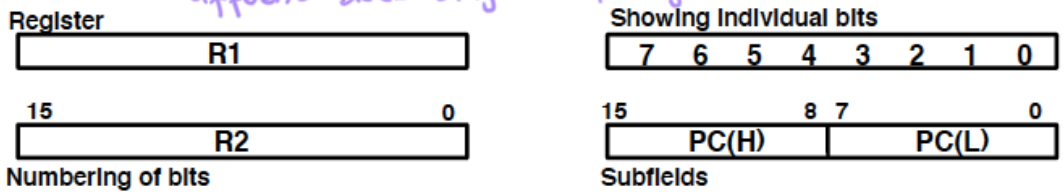
microoperations = the operations on the data in the registers.
↳ ex: shift, load, clear, arithmetic, logic

in transfer level language notation

List of BC Registers (basic)

DR	16	Data Register	Holds memory operand
AR	12	Address Register	Holds address for memory
AC	16	Accumulator	Processor register
IR	16	Instruction Register	Holds instruction code
PC	12	Program Counter	Holds address of instruction
TR	16	Temporary Register	Holds temporary data
INPR	8	Input Register	Holds input character
OUTR	8	Output Register	Holds output character

different block diagram of registers



organization of a digita system (computer) = registers + microoperations + control signals
in register tranfer level

Register Transfer Language

register transfer = R3 ← R5 → copying all the contents of one register to another
- during one clock pulse, parallel load in R3
- R5 remains same
- there one control lines to perform action

control functions = P: R3 ← R5
- if P=1, then load the contents of register R5 into R3.

Simultaneous operation = P: R3 ← R5, MAR ← IR

basic symbols for register transfers

Symbols	Description	Examples
Capital letters & numerals	Denotes a register	MAR, R2
Parentheses ()	Denotes a part of a register	R2(0-7), R2(L)
Arrow ←	Denotes transfer of information	R2 ← R1
Colon :	Denotes termination of control function	P:
Comma ,	Separates two micro-operations	A ← B, B ← A

connecting registers:

instead ↳ not practical to use n(n-1) lines to load each register with all other register conte

the bus = one centeralized set of circuits for data transfer

↳ have control circuits to select source and destination registers

↳ using decoders, mux es, three state buffers,

R2 ← R1 = BUS ← R1, R2 ← BUS

memory: (ram)

↳ a sequential circuit that contains some number of registers

M = memory, MAR / AR = memory address register

memory read: R1 ← M[MAR] → read a value from a location in memory and load it into a register

memory write: M[MAR] ← R1 → write a value from a register to a location in memory

microoperations in computer systems

1. register transfer
2. arithmetic
3. logic → 16 different logic functions between two bits
4. shift

A ← B	Transfer content of reg. B into reg. A
AR ← DR(AD)	Transfer content of AD portion of reg. DR into reg. AR
A ← constant	Transfer a binary constant into reg. A
ABUS ← R1, R2 ← ABUS	Transfer content of R1 into bus A and, at the same time, transfer content of bus A into R2
AR	Address register
DR	Data register
M[R]	Memory word specified by reg. R
M	Equivalent to M[AR]
DR ← M	Memory *read* operation: transfers content of memory word specified by AR into DR
M ← DR	Memory *write* operation: transfers content of DR into memory word specified by AR

① register transfer

Summary of Typical Arithmetic Micro-Operations

R3 ← R1 + R2	Contents of R1 plus R2 transferred to R3
R3 ← R1 - R2	Contents of R1 minus R2 transferred to R3
R2 ← R2'	Complement the contents of R2
R2 ← R2'+ 1	2's complement the contents of R2 (negate)
R3 ← R1 + R2'+ 1	subtraction
R1 ← R1 + 1	Increment
R1 ← R1 - 1	Decrement

② arithmetic

- Selective-set $A \leftarrow A + B$
- Selective-complement $A \leftarrow A \oplus B$
- Selective-clear $A \leftarrow A \cdot B'$
- Mask (Delete) $A \leftarrow A \cdot B$
- Clear $A \leftarrow A \oplus B$
- Insert $A \leftarrow (A \cdot B) + C$
- Compare $A \leftarrow A \oplus B$

③ logic

logic microoperations

selective set: $A \leftarrow A+B$ → if a bit in B is set to 1, that same position in A gets set to 1, otherwise that bit in A keeps its previous value

not addition!

set certain bits

ex: 1 1 0 0 A_t

1 0 1 0 B

1 1 1 0 $A_{(t+1)}$ $(A \leftarrow A+B)$

selective complement: $A \leftarrow A \oplus B$ → if a bit in B is set to 1, that same position in A gets complemented from its original value, otherwise it is unchanged. complement certain bi

ex: 1 1 0 0 A+
1 0 1 0 B
0 1 1 0 A(t+1) ($A \leftarrow A \oplus B$)

selective clear: $A \leftarrow A.B'$ → if a bit in B is set to 1, that position in A gets set to 0, otherwise it is unchanded clear certain bits

ex: 1 1 0 0 A+
1 0 1 0 B
0 1 0 0 A(t+1) ($A \leftarrow A.B'$)

mask operation: $A \leftarrow A.B$ → if a bit in B is set to 0, that same position in A gets (delete) to set 0, otherwise it is unchanged. clear certain bits

ex: 1 1 0 0 A+
1 0 1 0 B
1 0 0 0 A(t+1) ($A \leftarrow A.B$)

clear operation: $A \leftarrow A \oplus B$ → if the bits in the same position in A and B are the same, they are cleared in A, otherwise they are set in A.

ex: 1 1 0 0 A+
1 0 1 0 B
0 1 1 0 A(t+1) ($A \leftarrow A \oplus B$)

insert operation: $A \leftarrow (A.B)+C$ → to introduce a specific bit pattern into A register, leaving the other bit positions unchanged: done as:

↳ a mask operation to clear the desired bit positions

↳ an OR operation to introduce the new bits into the desired positions

ex: 1101 1000 1011 0001 → A original
1101 1000 1011 1010 → A desired

↳ 1101 1000 1011 0001 → A
1111 1111 1111 0000 → mask
1101 1000 1011 0000 → A (intermediate)
0000 0000 0000 1010 → added bits
1101 1000 1011 1010 → A desired

shift microoperations

① logical shift

the serial input to the shift is a 0. (right shift of 111 is 011, left shift 110)

shl: logical shift left $R2 \leftarrow shr\ R2$
shr: logical shift right $R3 \leftarrow shl\ R3$
} in a register transfer language

② circular shift

the serial input is the bit that shifted out of the other end of the register.

cil: circular shift left $R2 \leftarrow cil\ R2$
cir: circular shift right $R3 \leftarrow cir\ R3$

(1011 —cil→ 0111, 1011 —cir→ 1101)

③ arithmatic shift

meant for signed integers, (keeps the sign of the number the same as right/left shifts)

ashr: arithmetic shift right → divides a signed number by two (10100111 → 11010011)
ashl: arithmetic shift left → multiplies a signed number by two (10100111 → 01001110)

↳ $R_2 \leftarrow ashr\ R2$ $R_2 \leftarrow ashl\ R_2$

overflow = ash left shift must check, if the leftmost two bits differ → the shift will result in an over.