🟢 SECTION A — SWITCH CONTROLLED LOGIC SYSTEMS

(ATmega328P – Assembly – 2 Digit Multiplexed – Common Cathode)

==========

==========

PROGRAM 1 — ADD SUB MUL DIV2 — 2 SWITCH ARITHMETIC SELECT

Keywords: ADD SUB MUL DIV SHIFT DIV2 ARITHMETIC SWITCH

==========

==========

🧠 PROBLEM DESCRIPTION

Use 2 switches (SW0–SW1) to select arithmetic operation:

SW1 SW0	Operation
00	op1 + op2
01	op1 - op2

10	op1 × op2
11	op1 ÷ 2

Operands:

- op1 = upper nibble of PORTC
- op2 = lower nibble of PORTC

Display result (0–99) on 2-digit multiplexed 7-segment.

🔌 HARDWARE MAPPING

- PORTD → segments (a–g,dp)
- PORTB0 → digit1 enable (tens)
- PORTB1 → digit2 enable (ones)
- PORTC → DIP switches

🧾 REGISTER USAGE

Register	Purpose
r17	op1
r18	op2
r19	result
r20	switch
r21	tens
r22	ones

⚙️ ALGORITHM

1. Read PORTC
2. Extract switch bits
3. Extract operands
4. Perform selected arithmetic
5. Convert binary → decimal (divide by 10 loop)
6. Multiplex display

💻 FULL CODE

```
.include "m328pdef.inc"

.def temp = r16
.def op1  = r17
.def op2  = r18
.def res  = r19
.def sw   = r20
.def tens = r21
.def ones = r22

.org 0x00
rjmp start

seg_table:
.db 0x3F,0x06,0x5B,0x4F,0x66,0x6D,0x7D,0x07,0x7F,0x6F

start:
   ldi temp,0xFF
   out DDRD,temp
   ldi temp,0x03
   out DDRB,temp
   ldi temp,0x00
   out DDRC,temp

main:
```

```
    in temp,PINC
    mov sw,temp
    andi sw,0x03

    mov op1,temp
    andi op1,0xF0
    swap op1
    andi op1,0x0F

    mov op2,temp
    andi op2,0x0F

    cpi sw,0
    breq add_op
    cpi sw,1
    breq sub_op
    cpi sw,2
    breq mul_op
    rjmp div_op

add_op:
    mov res,op1
    add res,op2
    rjmp display

sub_op:
    mov res,op1
    sub res,op2
    rjmp display

mul_op:
    mul op1,op2
    mov res,r0
    clr r1
    rjmp display

div_op:
    mov res,op1
    lsr res
    rjmp display

display:
    ldi tens,0
divide10:
```

```
    cpi res,10
    brlo split_done
    subi res,10
    inc tens
    rjmp divide10

split_done:
    mov ones,res

display_loop:
    ldi ZH,high(seg_table<<1)
    ldi ZL,low(seg_table<<1)
    add ZL,tens
    adc ZH,__zero_reg__
    lpm temp,Z
    out PORTD,temp
    sbi PORTB,0
    rcall delay
    cbi PORTB,0

    ldi ZH,high(seg_table<<1)
    ldi ZL,low(seg_table<<1)
    add ZL,ones
    adc ZH,__zero_reg__
    lpm temp,Z
    out PORTD,temp
    sbi PORTB,1
    rcall delay
    cbi PORTB,1

    rjmp main

delay:
    ldi r23,200
d1: ldi r24,200
d2: dec r24
    brne d2
    dec r23
    brne d1
    ret
```

==========

==========

PROGRAM 2 — NAND NOR XOR XNOR — LOGIC SELECT

Keywords: NAND NOR XOR XNOR LOGIC SWITCH

==========

==========

🧠 DESCRIPTION

SW2–SW3 select logic operation:

SW3 SW2	Operation
00	NAND
01	NOR
10	XOR
11	XNOR

Operands = lower 4 bits & upper 4 bits.

🧮 LOGIC RULES

- NAND = NOT (A AND B)
- NOR = NOT (A OR B)
- XOR = A ⊕ B
- XNOR = NOT (A ⊕ B)

💻 OPERATION BLOCK

Replace arithmetic block with:

```
  andi sw,0x0C
   lsr sw
   lsr sw

   mov res,op1

   cpi sw,0
   breq nand_op
   cpi sw,1
   breq nor_op
   cpi sw,2
   breq xor_op
   rjmp xnor_op

nand_op:
   and res,op2
   com res
   rjmp display

nor_op:
   or res,op2
   com res
   rjmp display

xor_op:
   eor res,op2
   rjmp display

xnor_op:
```

```
    eor res,op2
    com res
    rjmp display
```

==========

==========

PROGRAM 3 — UP DOWN COUNTER

Keywords: UP COUNTER DOWN COUNTER SWITCH MODE

==========

==========

🧠 DESCRIPTION

SW0:

- 0 → Up counter
- 1 → Down counter

Counts continuously and displays value.

💻 MAIN LOOP

```
main:
    in sw,PINC
```

```
    sbrs sw,0
    rjmp up_mode

down_mode:
    dec res
    rjmp display

up_mode:
    inc res
    rjmp display
```

==========

==========

PROGRAM 4 — BINARY / GRAY CODE DISPLAY

Keywords: GRAY CODE BINARY CONVERSION

==========

==========

🧠 DESCRIPTION

SW0:

- 0 → Show binary

- 1 → Show Gray code

Gray formula:
Gray = Binary XOR (Binary >> 1)

💻 GRAY CONVERSION

```
gray_mode:
   mov temp,res
   lsr temp
   eor res,temp
   rjmp display
```

==========

==========

PROGRAM 5 — FORMULA SELECT (x+5, 2x, x^2, x^3)

Keywords: POLYNOMIAL SQUARE CUBE FORMULA SWITCH

==========

==========

🧠 DESCRIPTION

SW1 SW0 select formula:

SW	Formula
00	x+5
01	2x
10	x^2
11	x^3

x = lower nibble of PORTC

💻 FORMULA BLOCK

```
  mov res,op1

   cpi sw,0
   breq f1
   cpi sw,1
   breq f2
   cpi sw,2
   breq f3
   rjmp f4

f1:
   subi res,-5
   rjmp display

f2:
   lsl res
   rjmp display

f3:
   mul res,res
   mov res,r0
   clr r1
   rjmp display

f4:
   mul res,res
   mov res,r0
   clr r1
   mul res,op1
```

```
mov res,r0
clr r1
rjmp display
```

🔥 NEXT

I will continue:

- PROGRAM 6–10
- PROGRAM 11–20
- Then SECTION B

Say:

> Continue Section A (6–10)

And we build your full exam USB reference manual properly.

continue

Perfect 🔥
Continuing your **EXAM USB REFERENCE MANUAL**

Platform:

- ATmega328P
- AVR Assembly
- 2-digit multiplexed
- Common Cathode
- Same segment table + display routine as Program 1

(For all programs below: reuse the same seg_table, display, and delay routines from Program 1.)

==========

==========

PROGRAM 6 — ROL / ROR SELECT

Keywords: ROL ROR ROTATE LEFT ROTATE RIGHT SWITCH

==========

==========

🧠 DESCRIPTION

Switch SW0 selects rotation direction:

SW0	Operation
0	ROL (Rotate Left)
1	ROR (Rotate Right)

Input value = PORTC
Result displayed on 2-digit display.

⚙️ ALGORITHM

1. Read PORTC
2. Store into res
3. Check SW0

4. Apply ROL or ROR
5. Display result

💻 MAIN LOGIC

```
main:
   in res,PINC
   mov sw,res
   andi sw,0x01

   cpi sw,0
   breq do_rol

do_ror:
   ror res
   rjmp display

do_rol:
   rol res
   rjmp display
```

==========

==========

PROGRAM 7 — EVEN / ODD COUNTER SELECT

Keywords: EVEN COUNTER ODD COUNTER STEP 2

==========

==========

🧠 DESCRIPTION

SW0:

- 0 → Even counter (0,2,4,6…)
- 1 → Odd counter (1,3,5,7…)

⚙️ ALGORITHM

- Even mode → increment by 2 starting from 0
- Odd mode → increment by 2 starting from 1

💻 CODE

```
main:
   in sw,PINC
   andi sw,0x01

   cpi sw,0
   breq even_mode

odd_mode:
   cpi res,0
   brne odd_continue
   ldi res,1
odd_continue:
   subi res,-2
   rjmp display

even_mode:
```

```
subi res,-2
rjmp display
```

==========

==========

PROGRAM 8 — SPEED SELECT (FAST / MEDIUM / SLOW)

Keywords: DELAY SPEED CONTROL SWITCH

==========

==========

🧠 DESCRIPTION

SW1 SW0:

SW	Speed
00	Fast
01	Medium
10	Slow

Counter increments continuously.

💻 CODE

```
main:
   in sw,PINC
   andi sw,0x03

   inc res
   rcall display

   cpi sw,0
   breq fast

   cpi sw,1
   breq medium

slow:
   rcall delay
   rcall delay
   rcall delay
   rjmp main

medium:
   rcall delay
   rcall delay
   rjmp main

fast:
   rcall delay
   rjmp main
```

==========

==========

PROGRAM 9 — PORT SELECT (READ PORTB OR PORTC)

Keywords: PORT SELECT INPUT SWITCH SOURCE

==========

==========

🧠 DESCRIPTION

SW0:

- 0 → Read PORTC
- 1 → Read PORTB

Display selected port value.

💻 CODE

```
main:
   in sw,PINC
   andi sw,0x01

   cpi sw,0
   breq read_c

read_b:
   in res,PINB
   rjmp display

read_c:
   in res,PINC
   rjmp display
```

==========

==========

PROGRAM 10 — INVERT INPUT BEFORE DISPLAY

Keywords: INVERT COMPLEMENT INPUT SWITCH

==========

==========

🧠 DESCRIPTION

SW0:

- 0 → Normal display
- 1 → Inverted display (COM)

💻 CODE

```
main:
   in res,PINC
   mov sw,res
   andi sw,0x01

   cpi sw,0
   breq normal
```

```
invert:
   com res
   rjmp display

normal:
   rjmp display
```

==========

==========

PROGRAM 11 — FLASH TABLE A / B SELECT

Keywords: FLASH TABLE LOOKUP SELECT

==========

==========

🧠 DESCRIPTION

Two tables in Flash:

- Table A → 0–9
- Table B → 9–0

SW0 selects table.

💻 TABLES

```
tableA:
.db 0,1,2,3,4,5,6,7,8,9

tableB:
.db 9,8,7,6,5,4,3,2,1,0
```

💻 ACCESS CODE

```
  in sw,PINC
   andi sw,0x01

   ldi ZL,low(tableA<<1)
   ldi ZH,high(tableA<<1)

   cpi sw,0
   breq loadA

   ldi ZL,low(tableB<<1)
   ldi ZH,high(tableB<<1)

loadA:
   add ZL,res
   adc ZH,__zero_reg__
   lpm res,Z
   rjmp display
```

==========

==========

PROGRAM 12 — ARITHMETIC / LOGICAL MODE SELECT

Keywords: ARITHMETIC MODE LOGICAL MODE SWITCH

==========

==========

🧠 DESCRIPTION

SW0:

- 0 → Arithmetic (ADD)
- 1 → Logical (AND)

Operands = upper/lower nibble of PORTC

💻 CODE

```
  in temp,PINC

   mov op1,temp
   andi op1,0xF0
   swap op1
   andi op1,0x0F

   mov op2,temp
   andi op2,0x0F

   mov sw,temp
   andi sw,0x01

   cpi sw,0
   breq arithmetic

logical:
   mov res,op1
```

```
    and res,op2
    rjmp display

arithmetic:
    mov res,op1
    add res,op2
    rjmp display
```

==========

==========

PROGRAM 13 — SIGNED / UNSIGNED ADDITION

Keywords: SIGNED ADD UNSIGNED ADD OVERFLOW

==========

==========

🧠 DESCRIPTION

SW0:

- 0 → Unsigned ADD
- 1 → Signed ADD (check V flag)

💻 CODE

```
  mov res,op1
   add res,op2

   in sw,PINC
   andi sw,0x01
   cpi sw,0
   breq unsigned_mode

signed_mode:
   brvc no_overflow
   ldi res,99
no_overflow:
   rjmp display

unsigned_mode:
   rjmp display
```

==========

==========

PROGRAM 14 — BCD / BINARY ADDITION

Keywords: BCD ADD DA ADJUST

==========

==========

🧠 DESCRIPTION

SW0:

- 0 → Binary add
- 1 → BCD add (decimal adjust)

💻 CODE

```
mov res,op1
 add res,op2

 in sw,PINC
 andi sw,0x01
 cpi sw,0
 breq display

 cpi res,10
 brlo display
 subi res,-6
 rjmp display
```

==========

==========

PROGRAM 15 — LEFT SHIFT / RIGHT SHIFT

Keywords: LSL LSR SHIFT SWITCH

==========

==========

```
  in res,PINC
   mov sw,res
   andi sw,0x01

   cpi sw,0
   breq left_shift

right_shift:
   lsr res
   rjmp display

left_shift:
   lsl res
   rjmp display
```

🔥 Next I will continue:

- PROGRAM 16–20 (finish Section A completely)

Say:

Continue Section A (16–20)

And we complete Section A fully.

continue

Perfect 🔥
Now we finish **SECTION A (16–20)**
Same platform and assumptions.

Reuse:

- seg_table
- display
- delay
- Same port configuration as Program 1

==========

==========

PROGRAM 16 — 7-SEG OR LED BINARY OUTPUT SELECT

Keywords: 7SEG LED MODE SELECT OUTPUT TYPE

==========

==========

🧠 DESCRIPTION

SW0:

- 0 → Display decimal on 2-digit 7-seg
- 1 → Display raw binary on LEDs (PORTD)

Input = PORTC

⚙️ ALGORITHM

1. Read PORTC → res
2. Check SW0
3. If LED mode → output directly to PORTD

4. If 7-seg → convert to decimal and multiplex

💻 CODE

```
main:
   in res,PINC
   mov sw,res
   andi sw,0x01

   cpi sw,0
   breq seg_mode

led_mode:
   out PORTD,res
   rjmp main

seg_mode:
   rjmp display
```

==========

==========

PROGRAM 17 — MULTIPLY BY 2 / 4 / 8 SELECT

Keywords: MULTIPLY SHIFT LEFT LSL SCALE

==========

==========

🧠 DESCRIPTION

SW1 SW0:

SW	Operation
00	x ×2
01	x ×4
10	x ×8

x = PORTC

⚙️ SHIFT METHOD

Multiply by:

- 2 → LSL once
- 4 → LSL twice
- 8 → LSL three times

💻 CODE

```
main:
   in res,PINC
   mov sw,res
   andi sw,0x03

   cpi sw,0
   breq mul2
   cpi sw,1
```

```
    breq mul4
    rjmp mul8

mul2:
    lsl res
    rjmp display

mul4:
    lsl res
    lsl res
    rjmp display

mul8:
    lsl res
    lsl res
    lsl res
    rjmp display
```

==========

==========

PROGRAM 18 — SATURATE RESULT AT 99

Keywords: SATURATION LIMIT CLAMP MAX 99

==========

==========

🧠 DESCRIPTION

After arithmetic operation:

SW0:

- 0 → Normal
- 1 → If result > 99 → Force result = 99

⚙️ ALGORITHM

1. Compute result
2. Compare with 100
3. If ≥100 → load 99

💻 CODE (after arithmetic)

```
in sw,PINC
 andi sw,0x01
 cpi sw,0
 breq display

 cpi res,100
 brlo display

 ldi res,99
 rjmp display
```

==========

==========

PROGRAM 19 — DISPLAY CARRY FLAG

Keywords: CARRY FLAG CF STATUS REGISTER

==========

==========

🧠 DESCRIPTION

After addition:

SW0:

- 0 → Normal display
- 1 → If Carry=1 → Light LED (PB2)

⚙️ ALGORITHM

1. Perform ADD
2. Check C flag
3. If set → set PORTB2

💻 CODE

```
mov res,op1
 add res,op2
```

```
    in sw,PINC
    andi sw,0x01
    cpi sw,0
    breq display

    brcc no_carry
    sbi PORTB,2
    rjmp display

no_carry:
    cbi PORTB,2
    rjmp display
```

==========

==========

PROGRAM 20 — POINTER SELECT (X OR Y)

Keywords: POINTER X Y INDIRECT ADDRESSING SRAM

==========

==========

🧠 DESCRIPTION

SW0:

- 0 → Use X pointer
- 1 → Use Y pointer

Both point to different SRAM locations.

Load byte and display.

⚙️ POINTER REGISTERS

- X = r27:r26
- Y = r29:r28

💻 INITIALIZE POINTERS

```
  ldi r26,low(0x0100)
   ldi r27,high(0x0100)

   ldi r28,low(0x0110)
   ldi r29,high(0x0110)
```

💻 MAIN CODE

```
main:
   in sw,PINC
   andi sw,0x01

   cpi sw,0
   breq use_x

use_y:
   ld res,Y
   rjmp display

use_x:
```

```
ld res,X
rjmp display
```

✅ SECTION A COMPLETE (1–20)

You now have:

- Arithmetic select
- Logic select
- Up/down counter
- Gray code
- Formula select
- Rotate
- Even/odd
- Speed control
- Port select
- Invert
- Flash table select
- Arithmetic/logical mode
- Signed/unsigned
- BCD add
- Shift select
- 7-seg/LED select
- Multiply scale

- Saturation
- Carry detect
- Pointer select

🚀 NEXT

Now we move to:

🟢 SECTION B — COUNTER + 7-SEG SYSTEMS (21–40)

These are VERY IMPORTANT for exam.

Say:

Start Section B

And we continue building your full USB exam manual 🔥

next

Excellent 🔥
Now we begin:

🟢 SECTION B — COUNTER + 7-SEG SYSTEMS (21–40)

ATmega328P — Assembly — 2-digit multiplexed — Common Cathode

(Reuse seg_table, display, delay from Section A)

==========

==========

PROGRAM 21 — INCREMENT & RESET COUNTER (0–99)

Keywords: INCREMENT RESET 0-99 COUNTER

==========

==========

🧠 DESCRIPTION

Two switches:

- SW0 → Increment
- SW1 → Reset

Display range: 0–99

⚙️ ALGORITHM

1. If SW1 pressed → res = 0
2. If SW0 pressed → res++
3. If res = 100 → reset to 0

4. Display

💻 CODE

```
main:
   in sw,PINC

   sbrs sw,1
   rjmp check_inc
   clr res

check_inc:
   sbrs sw,0
   rjmp display

   inc res
   cpi res,100
   brlo display
   clr res
   rjmp display
```

==========

==========

PROGRAM 22 — COUNTER 0–59 (RESET AT 60)

Keywords: MOD 60 COUNTER

==========

==========

```
main:
   inc res
   cpi res,60
   brlo display
   clr res
   rjmp display
```

==========

==========

PROGRAM 23 — BIDIRECTIONAL COUNTER

Keywords: UP DOWN BIDIRECTIONAL

==========

==========

SW0:

- 0 → Up
- 1 → Down

```
main:
   in sw,PINC
   sbrs sw,0
```

```
    rjmp up

down:
    dec res
    rjmp display

up:
    inc res
    rjmp display
```

==========

==========

PROGRAM 24 — COUNTER INCREASES BY DIP VALUE

Keywords: STEP SIZE VARIABLE COUNTER

==========

==========

Step size = lower nibble of PORTC

```
main:
    in temp,PINC
    andi temp,0x0F
    add res,temp
    rjmp display
```

==========

==========

PROGRAM 25 — COUNTER STOPS WHEN SWITCH PRESSED

Keywords: PAUSE STOP SWITCH HOLD

==========

==========

SW0:

- 1 → Pause

```
main:
   in sw,PINC
   sbrs sw,0
   rjmp run

pause:
   rjmp display

run:
   inc res
   rjmp display
```

==========

==========

PROGRAM 26 — RESET ON OVERFLOW (8-BIT)

Keywords: OVERFLOW RESET

==========

==========

```
main:
    inc res
    brne display
    clr res
    rjmp display
```

==========

==========

PROGRAM 27 — EVEN NUMBERS ONLY (0–98)

Keywords: EVEN ONLY STEP 2

==========

==========

```
main:
```

```
    subi res,-2
    cpi res,100
    brlo display
    clr res
    rjmp display
```

==========

==========

PROGRAM 28 — PRIME COUNTER <100

Keywords: PRIME NUMBER COUNTER

==========

==========

🧠 DESCRIPTION

Display prime numbers less than 100.

💻 SIMPLE PRIME CHECK (2–97)

```
next_num:
    inc res
    cpi res,100
    brlo check_prime
    ldi res,2

check_prime:
    ldi r23,2
```

```
prime_loop:
    mov r24,res
    cp r23,res
    breq prime_found

div_loop:
    sub r24,r23
    brcc div_loop
    breq not_prime

    inc r23
    cp r23,res
    brlo prime_loop

prime_found:
    rjmp display

not_prime:
    rjmp next_num
```

(Exam note: This is brute force subtraction division.)

==========

==========

PROGRAM 29 — COUNT FROM DIP VALUE TO 0

Keywords: COUNTDOWN FROM INPUT

==========

==========

```
main:
   in sw,PINC
   mov res,sw

countdown:
   dec res
   brpl display
   rjmp main
```

==========

==========

PROGRAM 30 — PAUSE WHEN SWITCH HELD

Keywords: HOLD SWITCH PAUSE

==========

==========

```
main:
   in sw,PINC
   sbrs sw,0
   rjmp run

pause:
   rjmp display
```

```
run:
    inc res
    rjmp display
```

==========

==========

PROGRAM 31 — RISING EDGE INCREMENT

Keywords: RISING EDGE DETECT

==========

==========

Uses previous state register r25

```
main:
    in sw,PINC
    mov temp,sw
    andi temp,1

    cp temp,r25
    breq no_change

    cpi temp,1
    breq rising

no_change:
    mov r25,temp
    rjmp display
```

```
rising:
   inc res
   mov r25,temp
   rjmp display
```

==========

==========

PROGRAM 32 — FALLING EDGE DECREMENT

Keywords: FALLING EDGE DETECT

==========

==========

```
main:
   in sw,PINC
   mov temp,sw
   andi temp,1

   cp temp,r25
   breq no_change

   cpi temp,0
   breq falling

no_change:
   mov r25,temp
   rjmp display

falling:
   dec res
```

```
    mov r25,temp
    rjmp display
```

==========

==========

PROGRAM 33 — 0–255 DISPLAY LOW BYTE

Keywords: 8-BIT COUNTER

==========

==========

```
main:
    inc res
    rjmp display
```

(8-bit auto wrap)

==========

==========

PROGRAM 34 — 00–99 MULTIPLEXED DISPLAY

Keywords: 2 DIGIT MULTIPLEX

==========

==========

Just reuse:

```
inc res
cpi res,100
brlo display
clr res
rjmp display
```

(Display routine already multiplexed.)

==========

==========

PROGRAM 35 — 000–255 (3-DIGIT)

Keywords: 3 DIGIT MULTIPLEX

==========

==========

Add hundreds:

```
ldi r23,0
hundreds:
   cpi res,100
   brlo tens_calc
   subi res,100
   inc r23
   rjmp hundreds
```

Then continue divide10.

==========

==========

PROGRAM 36 — INCREASE EVERY 500ms

Keywords: TIMER DELAY 500ms

==========

==========

Call delay multiple times:

```
inc res
rcall delay
rcall delay
rcall delay
```

```
rcall delay
rcall delay
rjmp display
```

==========

==========

PROGRAM 37 — EXTERNAL CLOCK INPUT

Keywords: EXTERNAL CLOCK PIN CHANGE

==========

==========

```
sbis PINC,0
rjmp main

inc res
rjmp display
```

==========

==========

PROGRAM 38 — BINARY & DECIMAL SIMULTANEOUS

Keywords: BINARY DECIMAL DISPLAY

==========

==========

- PORTD → binary
- 7seg → decimal

out PORTD,res
rjmp display

==========

==========

PROGRAM 39 — RUNNING SUM

Keywords: ACCUMULATOR SUM

==========

==========

in temp,PINC

```
add res,temp
rjmp display
```

==========

==========

PROGRAM 40 — DISPLAY BCD FORMAT

Keywords: BCD DISPLAY

==========

==========

Assume res already BCD:

```
mov tens,res
andi tens,0xF0
swap tens
andi res,0x0F
mov ones,res
```

Then multiplex.

✅ SECTION B COMPLETE (21–40)

You now covered:

- Basic counters

- Edge detection
- Prime logic
- Pause control
- Variable step
- Multiplex
- Running sum
- BCD

Next:

🟢 SECTION C — BIT LOGICAL SYSTEMS (41–60)

These are very common old-exam style.

Say:

> Continue Section C

And we continue 🔥

next

Excellent 🔥
Now we continue:

🟢 SECTION C — BIT LOGICAL SYSTEMS (41–60)

ATmega328P — Assembly — 2-digit multiplexed — Common Cathode

Reuse:

- seg_table
- display
- delay

==========

==========

PROGRAM 41 — NAND NOR XOR XNOR (DISPLAY RESULT)

Keywords: NAND NOR XOR XNOR BITWISE

==========

==========

🧠 DESCRIPTION

PORTC upper nibble = A
PORTC lower nibble = B
SW1 SW0 select logic operation.

💻 CODE

```
in temp,PINC

mov op1,temp
andi op1,0xF0
swap op1
andi op1,0x0F

mov op2,temp
andi op2,0x0F

mov sw,temp
andi sw,0x03

mov res,op1

cpi sw,0
breq nand_op
cpi sw,1
breq nor_op
cpi sw,2
breq xor_op
rjmp xnor_op

nand_op:
and res,op2
com res
rjmp display

nor_op:
or res,op2
com res
rjmp display

xor_op:
eor res,op2
rjmp display

xnor_op:
eor res,op2
com res
rjmp display
```

==========

==========

PROGRAM 42 — LOGICAL COMPARATOR (A>B, A=B, A<B)

Keywords: COMPARATOR GREATER EQUAL LESS

==========

==========

🧠 DESCRIPTION

Display:

- 1 → A>B
- 2 → A=B
- 3 → A<B

💻 CODE

```
cp op1,op2
breq equal
brlo less

greater:
ldi res,1
```

```
rjmp display

equal:
ldi res,2
rjmp display

less:
ldi res,3
rjmp display
```

==========

==========

PROGRAM 43 — 2-BIT ALU SYSTEM

Keywords: MINI ALU 2BIT

==========

==========

Operations:

- 00 → ADD
- 01 → SUB
- 10 → AND
- 11 → OR

```
andi op1,0x03
andi op2,0x03
```

```
mov res,op1

cpi sw,0
breq add
cpi sw,1
breq sub
cpi sw,2
breq and_op
rjmp or_op

add:
add res,op2
rjmp display

sub:
sub res,op2
rjmp display

and_op:
and res,op2
rjmp display

or_op:
or res,op2
rjmp display
```

==========

==========

PROGRAM 44 — PARITY CHECKER

Keywords: PARITY EVEN ODD BIT COUNT

==========

==========

🧠 DESCRIPTION

If number of 1 bits is even → display 0
If odd → display 1

💻 CODE

```
clr r23
ldi r24,8

parity_loop:
lsr res
brcc skip
inc r23
skip:
dec r24
brne parity_loop

andi r23,1
mov res,r23
rjmp display
```

==========

==========

PROGRAM 45 — COUNT HIGH BITS

Keywords: POPCOUNT BIT COUNT

==========

==========

```
clr r23
ldi r24,8

count_loop:
lsr res
brcc no_inc
inc r23
no_inc:
dec r24
brne count_loop

mov res,r23
rjmp display
```

==========

==========

PROGRAM 46 — PALINDROME BIT PATTERN

Keywords: BIT PALINDROME MIRROR CHECK

==========

==========

Check if MSB mirror LSB.

```
mov r23,res
clr r24
ldi r25,8

reverse_loop:
lsl r23
ror r24
dec r25
brne reverse_loop

cp r24,res
breq yes

ldi res,0
rjmp display

yes:
ldi res,1
rjmp display
```

==========

==========

PROGRAM 47 — EXTRACT UPPER NIBBLE

Keywords: UPPER NIBBLE MASK

==========

==========

```
andi res,0xF0
swap res
andi res,0x0F
rjmp display
```

==========

==========

PROGRAM 48 — SWAP NIBBLES

Keywords: SWAP NIBBLE

==========

==========

```
swap res
rjmp display
```

==========

==========

PROGRAM 49 — ROTATE UNTIL MSB=1

Keywords: ROTATE UNTIL MSB SET

==========

==========

```
check_msb:
sbrc res,7
rjmp display

rol res
rjmp check_msb
```

==========

==========

PROGRAM 50 — BITWISE AND (PORTB & PORTC)

Keywords: AND TWO PORTS

==========

==========

```
in r23,PINB
in r24,PINC
and r23,r24
mov res,r23
rjmp display
```

==========

==========

PROGRAM 51 — BITWISE OR

Keywords: OR TWO PORTS

==========

==========

```
in r23,PINB
in r24,PINC
or r23,r24
mov res,r23
rjmp display
```

==========

==========

PROGRAM 52 — XOR THEN INVERT

Keywords: XOR INVERT

==========

==========

```
in r23,PINB
in r24,PINC
eor r23,r24
com r23
mov res,r23
rjmp display
```

==========

==========

PROGRAM 53 — MIRROR BIT PATTERN

Keywords: REVERSE BITS

==========

==========

(Same reverse logic as 46 but output reversed)

```
mov r23,res
clr r24
ldi r25,8

rev_loop:
lsl r23
ror r24
dec r25
brne rev_loop

mov res,r24
rjmp display
```

==========

==========

PROGRAM 54 — REVERSE LOWER NIBBLE

Keywords: LOWER NIBBLE REVERSE

==========

==========

```
andi res,0x0F
mov r23,res
clr r24
ldi r25,4

rev4:
lsl r23
ror r24
dec r25
brne rev4

mov res,r24
rjmp display
```

==========

==========

PROGRAM 55 — DETECT OVERFLOW AFTER ADD

Keywords: OVERFLOW FLAG V

==========

==========

```
add op1,op2
brvc no_over
ldi res,1
rjmp display

no_over:
ldi res,0
rjmp display
```

==========

==========

PROGRAM 56 — DETECT CARRY AND LIGHT LED

Keywords: CARRY FLAG CF

==========

==========

```
add op1,op2
brcc no_c
sbi PORTB,2
rjmp display

no_c:
cbi PORTB,2
rjmp display
```

==========

==========

PROGRAM 57 — DETECT HALF CARRY

Keywords: HALF CARRY H FLAG

==========

==========

```
add op1,op2
in r23,SREG
sbrc r23,5
ldi res,1
sbrs r23,5
ldi res,0
rjmp display
```

==========

==========

PROGRAM 58 — MASK EVEN BITS ONLY

Keywords: MASK 0x55 EVEN BITS

==========

==========

```
andi res,0x55
rjmp display
```

==========

==========

PROGRAM 59 — WALKING 1 PATTERN

Keywords: WALKING ONE

==========

==========

```
ldi res,1

loop:
```

```
lsl res
brne show
ldi res,1

show:
rjmp display
```

==========

==========

PROGRAM 60 — ALTERNATING 0xAA / 0x55

Keywords: ALTERNATE PATTERN TOGGLE

==========

==========

```
ldi res,0xAA

loop:
com res
rjmp display
```

✅ SECTION C COMPLETE (41–60)

You now covered:

- Bit logic
- Parity
- Bit count
- Bit reverse
- Comparator
- Flags (C, V, H)
- Masking
- Walking patterns

Next:

🟢 SECTION D — FLASH / SRAM / POINTER SYSTEMS (61–80)

These are memory-heavy exam questions.

Say:

> Continue Section D

And we proceed 🔥

next

Excellent 🔥
Now we move to:

🟢 SECTION D — FLASH / SRAM / POINTER SYSTEMS (61–80)

ATmega328P — AVR Assembly

Focus:

- LPM (Flash read)
- LD/ST (SRAM access)
- X, Y, Z pointers
- Multibyte arithmetic

==========

==========

PROGRAM 61 — COPY FLASH ARRAY TO SRAM + COUNT BYTES

Keywords: FLASH TO SRAM COPY LPM COUNT

==========

==========

🧠 DESCRIPTION

Copy Flash array to SRAM until 0xFF marker.
Count number of bytes copied. Display count.

💻 FLASH DATA

```
flash_array:
.db 10,20,30,40,50,0xFF
```

💻 CODE

```
ldi ZL,low(flash_array<<1)
ldi ZH,high(flash_array<<1)

ldi XL,low(0x0100)
ldi XH,high(0x0100)

clr res

copy_loop:
lpm r23,Z+
cpi r23,0xFF
breq done
st X+,r23
inc res
rjmp copy_loop

done:
rjmp display
```

==========

==========

PROGRAM 62 — COPY STRING + DISPLAY LENGTH

Keywords: STRING LENGTH COUNT

==========

==========

```
string:
.db "HELLO",0

ldi ZL,low(string<<1)
ldi ZH,high(string<<1)

clr res

len_loop:
lpm r23,Z+
cpi r23,0
breq done
inc res
rjmp len_loop

done:
rjmp display
```

==========

==========

PROGRAM 63 — REVERSE STRING IN SRAM

Keywords: STRING REVERSE POINTER

==========

==========

```
ldi XL,low(0x0100)
ldi XH,high(0x0100)

ldi YL,low(0x0110)
ldi YH,high(0x0110)

reverse_loop:
ld r23,X+
st -Y,r23
rjmp reverse_loop
```

(Exam: assume fixed length)

==========

==========

PROGRAM 64 — FIND MAX IN FLASH ARRAY

Keywords: MAXIMUM FIND

==========

==========

```
ldi ZL,low(flash_array<<1)
ldi ZH,high(flash_array<<1)
```

```
lpm res,Z+

max_loop:
lpm r23,Z+
cpi r23,0xFF
breq done
cp r23,res
brlo max_loop
mov res,r23
rjmp max_loop

done:
rjmp display
```

==========

==========

PROGRAM 65 — FIND MIN IN FLASH ARRAY

Keywords: MINIMUM FIND

==========

==========

```
lpm res,Z+

min_loop:
lpm r23,Z+
cpi r23,0xFF
breq done
cp r23,res
brsh min_loop
```

```
mov res,r23
rjmp min_loop
```

==========

==========

PROGRAM 66 — SUM 10 BYTES IN FLASH

Keywords: SUM ARRAY

==========

==========

```
ldi r24,10
clr res

sum_loop:
lpm r23,Z+
add res,r23
dec r24
brne sum_loop

rjmp display
```

==========

==========

PROGRAM 67 — AVERAGE 10 NUMBERS

Keywords: AVERAGE DIVIDE

==========

==========

```
; assume sum already in res
ldi r23,10
div_loop:
sub res,r23
brcc div_loop
add res,r23
rjmp display
```

(Simple repeated subtraction divide)

==========

==========

PROGRAM 68 — COUNT EVEN NUMBERS

Keywords: EVEN COUNT ARRAY

==========

==========

```
clr res
```

```
ldi r24,10

loop:
lpm r23,Z+
andi r23,1
brne skip
inc res
skip:
dec r24
brne loop

rjmp display
```

==========

==========

PROGRAM 69 — REPLACE CHARACTER IN STRING

Keywords: STRING MODIFY

==========

==========

Replace 'A' with 'Z'

```
ldi XL,low(0x0100)
ldi XH,high(0x0100)

loop:
ld r23,X
cpi r23,0
```

```
breq done
cpi r23,'A'
brne next
ldi r23,'Z'
st X,r23
next:
adiw XL,1
rjmp loop

done:
rjmp display
```

==========

==========

PROGRAM 70 — COMPARE TWO FLASH STRINGS

Keywords: STRING COMPARE

==========

==========

```
compare_loop:
lpm r23,Z+
lpm r24,Y+
cp r23,r24
brne not_equal
cpi r23,0
breq equal
rjmp compare_loop

equal:
```

```
ldi res,1
rjmp display

not_equal:
ldi res,0
rjmp display
```

==========

==========

PROGRAM 71 — ADD TWO ARRAYS

Keywords: ARRAY ADD

==========

==========

```
ldi r24,10

loop:
ld r23,X+
ld r25,Y+
add r23,r25
st Z+,r23
dec r24
brne loop
```

==========

==========

PROGRAM 72 — MULTIBYTE ADD (4 BYTES)

Keywords: 32BIT ADD ADC

==========

==========

```
clc
ldi r24,4

loop:
ld r23,X+
ld r25,Y+
adc r23,r25
st Z+,r23
dec r24
brne loop
```

==========

==========

PROGRAM 73 — BCD MULTIBYTE ADD

Keywords: BCD ADC ADJUST

==========

==========

```
add r23,r25
cpi r23,10
brlo store
subi r23,-6
store:
st Z+,r23
```

==========

==========

PROGRAM 74 — SUBTRACT 4-BYTE NUMBERS

Keywords: MULTIBYTE SUB SBC

==========

==========

```
clc
ldi r24,4

loop:
ld r23,X+
ld r25,Y+
sbc r23,r25
st Z+,r23
dec r24
brne loop
```

==========

==========

PROGRAM 75 — MULTIPLY 8-BIT FROM MEMORY

Keywords: MUL MEMORY

==========

==========

```
ld r23,X
ld r24,Y
mul r23,r24
mov res,r0
clr r1
rjmp display
```

==========

==========

PROGRAM 76 — SEARCH VALUE IN FLASH

Keywords: SEARCH ARRAY

==========

==========

```
ldi r25,50

search:
lpm r23,Z+
cpi r23,0xFF
breq not_found
cp r23,r25
breq found
rjmp search

found:
ldi res,1
rjmp display

not_found:
ldi res,0
rjmp display
```

==========

==========

PROGRAM 77 — TRANSFER ODD INDEX BYTES

Keywords: ODD INDEX COPY

==========

==========

```
ldi r24,10
clr r25

loop:
lpm r23,Z+
andi r25,1
brne store
rjmp skip
store:
st X+,r23
skip:
inc r25
dec r24
brne loop
```

==========

==========

PROGRAM 78 — FIND INDEX OF LARGEST ELEMENT

Keywords: MAX INDEX

==========

==========

```
clr r26
clr r27
```

```
lpm res,Z+

loop:
lpm r23,Z+
cpi r23,0xFF
breq done
inc r27
cp r23,res
brlo loop
mov res,r23
mov r26,r27
rjmp loop

done:
mov res,r26
rjmp display
```

==========

==========

PROGRAM 79 — TRANSFER UNTIL SPECIAL MARKER

Keywords: COPY UNTIL 0x00

==========

==========

```
loop:
lpm r23,Z+
cpi r23,0
breq done
```

```
st X+,r23
rjmp loop

done:
rjmp display
```

==========

==========

PROGRAM 80 — DOUBLE EACH ELEMENT STORE TO SRAM

Keywords: SCALE ARRAY

==========

==========

```
ldi r24,10

loop:
lpm r23,Z+
lsl r23
st X+,r23
dec r24
brne loop
```

✅ SECTION D COMPLETE (61–80)

You now mastered:

- LPM
- Flash arrays
- String processing
- Multibyte add/sub
- Pointer arithmetic
- Searching
- Index tracking

Next:

🟢 SECTION E — ARITHMETIC + DISPLAY SYSTEMS (81–95)

These combine math + display logic.

Say:

> Continue Section E

And we continue 🔥

next

Perfect 🔥
Now we continue:

🟢 SECTION E — ARITHMETIC + DISPLAY SYSTEMS (81–95)

ATmega328P — AVR Assembly — 2-digit multiplexed

(Reuse seg_table, display, delay)

==========

==========

PROGRAM 81 — y = 3x + 7

Keywords: LINEAR EQUATION 3X+7

==========

==========

🧠 DESCRIPTION

x = PORTC
Compute: y = 3x + 7
Display result.

💻 CODE

```
in res,PINC
mov r23,res
lsl res
add res,r23
subi res,-7
rjmp display
```

==========

==========

PROGRAM 82 — y = x² + 2x + 9

Keywords: QUADRATIC EQUATION

==========

==========

```
in r23,PINC
mov r24,r23

mul r23,r23
mov res,r0
clr r1

lsl r24
add res,r24
subi res,-9

rjmp display
```

==========

==========

PROGRAM 83 — MULTIPLY TWO SWITCH INPUTS

Keywords: MUL TWO INPUTS

==========

==========

Upper nibble = A
 Lower nibble = B

```
in temp,PINC

mov op1,temp
andi op1,0xF0
swap op1
andi op1,0x0F

mov op2,temp
andi op2,0x0F

mul op1,op2
mov res,r0
clr r1

rjmp display
```

==========

==========

PROGRAM 84 — DIVIDE SUM BY 2

Keywords: AVERAGE DIV2

==========

==========

```
add op1,op2
lsr op1
mov res,op1
rjmp display
```

==========

==========

PROGRAM 85 — FACTORIAL (SMALL NUMBER)

Keywords: FACTORIAL LOOP

==========

==========

```
x = lower nibble (0–5 safe)

in res,PINC
andi res,0x0F

mov r23,res
ldi r24,1

fact_loop:
mul r24,r23
mov r24,r0
clr r1
```

```
dec r23
brne fact_loop

mov res,r24
rjmp display
```

==========

==========

PROGRAM 86 — FIBONACCI NUMBER

Keywords: FIBONACCI SERIES

==========

==========

n = lower nibble

```
in r23,PINC
andi r23,0x0F

ldi r24,0
ldi r25,1

fib_loop:
cp r23,0
breq done
mov r26,r25
add r25,r24
mov r24,r26
dec r23
rjmp fib_loop
```

```
done:
mov res,r24
rjmp display
```

==========

==========

PROGRAM 87 — ADD TWO 4-BYTE NUMBERS (SHOW LOW BYTE)

Keywords: 32BIT ADD DISPLAY LOW

==========

==========

```
clc
ldi r24,4

loop:
ld r23,X+
ld r25,Y+
adc r23,r25
st Z+,r23
dec r24
brne loop

mov res,r23
rjmp display
```

==========

==========

PROGRAM 88 — DISPLAY HIGH BYTE OF MULTIPLICATION

Keywords: MUL HIGH BYTE

==========

==========

```
mul op1,op2
mov res,r1
clr r1
rjmp display
```

==========

==========

PROGRAM 89 — BINARY TO BCD

Keywords: BIN TO BCD CONVERSION

==========

==========

Simple repeated subtraction:

```
clr tens

bin_loop:
cpi res,10
brlo done
subi res,10
inc tens
rjmp bin_loop

done:
mov ones,res
```

Then multiplex tens & ones.

==========

==========

PROGRAM 90 — BCD TO BINARY

Keywords: BCD TO BIN

==========

==========

```
mov tens,res
andi tens,0xF0
```

```
swap tens

andi res,0x0F

ldi r23,10
mul tens,r23
add r0,res

mov res,r0
clr r1
rjmp display
```

==========

==========

PROGRAM 91 — DETECT OVERFLOW AND BLINK LED

Keywords: OVERFLOW BLINK

==========

==========

```
add op1,op2
brvc normal

overflow:
sbi PORTB,2
rcall delay
cbi PORTB,2
rjmp display

normal:
```

```
rjmp display
```

==========

==========

PROGRAM 92 — SATURATED ADDITION

Keywords: SATURATION ADD CLAMP 255

==========

==========

```
add op1,op2
brcc no_sat
ldi res,255
rjmp display

no_sat:
mov res,op1
rjmp display
```

==========

==========

PROGRAM 93 — ABSOLUTE DIFFERENCE

Keywords: ABS DIFFERENCE

==========

==========

```
cp op1,op2
brlo swap

sub op1,op2
mov res,op1
rjmp display

swap:
sub op2,op1
mov res,op2
rjmp display
```

==========

==========

PROGRAM 94 — MULTIPLY USING SHIFT-ADD METHOD

Keywords: SHIFT ADD MULTIPLY

==========

==========

```
clr res
```

```
ldi r24,8

mul_loop:
sbrc op2,0
add res,op1

lsl op1
lsr op2

dec r24
brne mul_loop

rjmp display
```

==========

==========

PROGRAM 95 — POLYNOMIAL USING LOOKUP TABLE

Keywords: LOOKUP TABLE POLYNOMIAL

==========

==========

Flash table:

```
poly_table:
.db 5,9,15,23,33,45,59,75

in res,PINC
andi res,0x07
```

```
ldi ZL,low(poly_table<<1)
ldi ZH,high(poly_table<<1)

add ZL,res
adc ZH,__zero_reg__
lpm res,Z

rjmp display
```

✅ SECTION E COMPLETE (81–95)

You now covered:

- Linear & quadratic equations
- Factorial
- Fibonacci
- 32-bit arithmetic
- Saturation
- Absolute value
- Shift-add multiplication
- Lookup tables
- Binary/BCD conversions

🔥 FINAL SECTION REMAINING:

🟢 SECTION F — AJARN STYLE COMBINED PROBLEMS (96–100)

These are high probability exam patterns.

Say:

Finish Section F

And we complete your full 100-program USB reference manual.

yes

Perfect 😈🔥
Now we finish your full exam arsenal.

🟢 SECTION F — COMBINED "AJARN STYLE" PROBLEMS (96–100)

ATmega328P — AVR Assembly — 2-digit multiplexed 7-seg

These are **very high probability exam style** questions.

==========

==========

PROGRAM 96 — SWITCH SELECTS OPERATION → 2-DIGIT DISPLAY

Keywords: SELECT OPERATION + MULTIPLEX

==========

==========

🧠 DESCRIPTION

- SW0–1 → Select operation
    - 00 = ADD
    - 01 = SUB
    - 10 = MUL
    - 11 = DIV2
- SW2–5 → Operand A
- SW6–7 → Operand B
- Display result on 2-digit 7-seg.

💻 CODE

```
in temp,PINC

; Extract operation
mov r20,temp
andi r20,0x03

; Operand A (bits2-5)
mov op1,temp
andi op1,0x3C
lsr op1
```

```
lsr op1

; Operand B (bits6-7)
mov op2,temp
andi op2,0xC0
swap op2
andi op2,0x03

cpi r20,0
breq add_case
cpi r20,1
breq sub_case
cpi r20,2
breq mul_case

; DIV2
lsr op1
mov res,op1
rjmp display

add_case:
add op1,op2
mov res,op1
rjmp display

sub_case:
sub op1,op2
mov res,op1
rjmp display

mul_case:
mul op1,op2
mov res,r0
clr r1
rjmp display
```

==========

==========

PROGRAM 97 — DIP → SQUARE → DISPLAY → BLINK IF >50

Keywords: SQUARE + BLINK CONDITION

==========

==========

🧠 DESCRIPTION

- Read DIP (0–15)
- Compute x^2
- If result > 50 → blink LED
- Always display result

💻 CODE

```
in op1,PINC
andi op1,0x0F

mul op1,op1
mov res,r0
clr r1

cpi res,51
brlo normal

; Blink LED
sbi PORTB,2
rcall delay
cbi PORTB,2
```

normal:
rjmp display

==========

==========

PROGRAM 98 — COPY FLASH ARRAY → SUM → DISPLAY

Keywords: FLASH SUM DISPLAY

==========

==========

🧠 DESCRIPTION

- Flash contains 10 bytes
- Copy to SRAM
- Sum all
- Display result

💻 CODE

Flash:

```
array:
.db 5,3,8,2,9,4,7,1,6,10


Main:

ldi ZL,low(array<<1)
ldi ZH,high(array<<1)

ldi XL,0x60
ldi XH,0x00

ldi r24,10
clr res

loop:
lpm r23,Z+
st X+,r23
add res,r23
dec r24
brne loop

rjmp display
```

==========

==========

PROGRAM 99 — COUNTER → APPLY ARITHMETIC → DISPLAY

Keywords: COUNTER + PROCESS

🧠 DESCRIPTION

- Counter increments
- Before display → compute y = 2x + 3
- Show on 2-digit

💻 CODE

```
ldi res,0

main:
inc res

mov r23,res
lsl r23
subi r23,-3

mov res,r23

rcall display
rcall delay
rjmp main
```

PROGRAM 100 — MINI ALU SYSTEM (FULL EXAM STYLE)

Keywords: MINI ALU COMPLETE SYSTEM

==========

==========

🧠 DESCRIPTION

- SW0–1 = Operand A
- SW2–3 = Operand B
- SW4–5 = Operation
    - 00 = ADD
    - 01 = SUB
    - 10 = AND
    - 11 = OR
- Result → LEDs (PORTD)
- Operation symbol displayed on 7-seg
- Includes delay subroutine

🔥 This is VERY typical Ajarn question.

💻 CODE

```
; Read switches
in temp,PINC

; Operand A
mov op1,temp
andi op1,0x03

; Operand B
mov op2,temp
andi op2,0x0C
lsr op2
lsr op2

; Operation select
mov r20,temp
andi r20,0x30
lsr r20
lsr r20
lsr r20
lsr r20

cpi r20,0
breq add_case
cpi r20,1
breq sub_case
cpi r20,2
breq and_case

; OR
or op1,op2
mov res,op1
rjmp output

add_case:
add op1,op2
mov res,op1
rjmp output

sub_case:
sub op1,op2
mov res,op1
rjmp output

and_case:
```

```
and op1,op2
mov res,op1

output:
out PORTD,res      ; LEDs show result

; Display operation number (0–3)
mov res,r20
rcall display

rjmp main
```