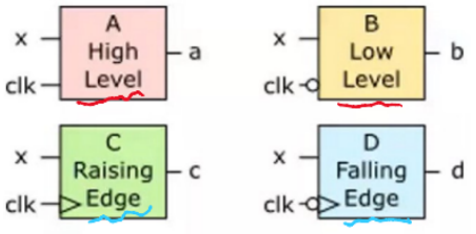
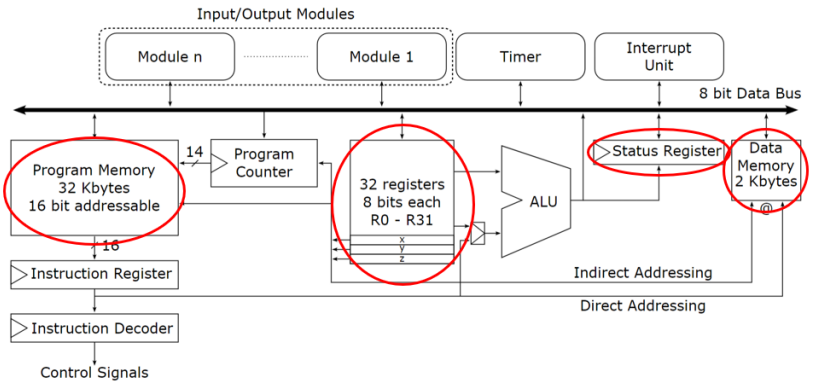
ELEC1601 A4 Exam Notes
530427410

Name	AND Form	OR Form
Identity Law	$1A = A$	$0 + A = A$
Null Law	$0A = 0$	$1 + A = 1$
Idempotent Law	$AA = A$	$A + A = A$
Inverse Law	$A\bar{A} = 0$	$A + \bar{A} = 1$
Commutative Law	$AB = BA$	$A + B = B + A$
Associative Law	$(AB)C = A(BC)$	$(A + B) + C = A + (B + C)$
Distributive Law	$A + BC = (A + B)(A + C)$	$A(B + C) = AB + AC$
Absorption Law	$A(A + B) = A$	$A + AB = A$
De Morgan's Law	$\overline{AB} = \bar{A} + \bar{B}$	$\overline{(A+B)} = \bar{A}\bar{B}$

SOP:
(1) Fetch all rows with output one
(2) Negate a variable if it's zero

2s COMP: if and only if negative
(1) Invert the bits
(2) Add 1





Binary range: $[0, (2^n - 1)]$

$$b = log_2(n) = \frac{log\ n}{log\ 2}$$

$$n = 2^b$$

Canonical design flow:
(a) Circuit → Truth table → Algebra (SOP)
(b) Algebra → Truth table → SOP → Circuit

Memory size: $total = 2^N \times M$

Frequency in clock: $f = \frac{1}{T}$

Worst-case rounding error: Halve the LSB

🚩**(C)** Carry flag, **(Z)** Zero flag, **(N)** Negative flag

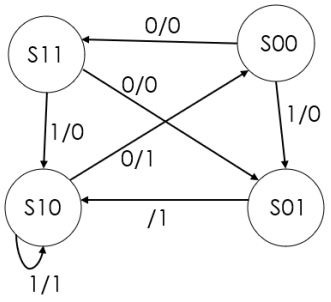
(X) r27:r26, **(Y)** r29:r28, **(Z)** r31:r30

Program memory: flash mem, 2 bytes/cell, 14 bit address (so has 2^14 locations), total 32 KB.

Data memory: Static RAM chip, 1 byte each cell, total 2KB.

ALU: Performs operations of 3 types: arithmetic, logical, bit level.

S_0	S_1	i	S_0^+	S_1^+	o
0	0	0	1	1	0
0	0	1	0	1	0
0	1	0	1	0	1
0	1	1	1	0	1
1	0	0	0	0	1
1	0	1	1	0	1
1	1	0	0	1	0
1	1	1	1	0	0



AVR starter template

```
.section .data
        ; insert data definitions here
.section .text
        .global
asm_function
asm_function:
        ; insert code here
        ret
.end
```

AVR if/else

```
…
; compare
; branch to `if`
else:
        …
        jmp
after_if
if:
        …
after_if:
        …
```

AVR for loop

```
…
loop_start:
        ; init values
check:
        ; compare
        ; br.
after_loop
loop:
        …
        jmp check
after_loop:
        …
```

AVR switch

```
…
        ; do maths
        ; check which label to branch
default
        …
        jmp after_switch
case1:
        …
        jmp after_switch
; case… until n-th case
after_switch:
        …
```