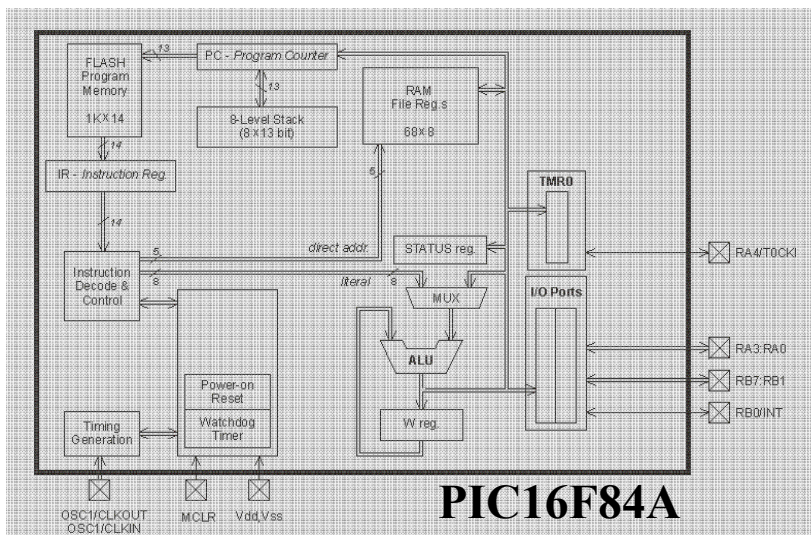
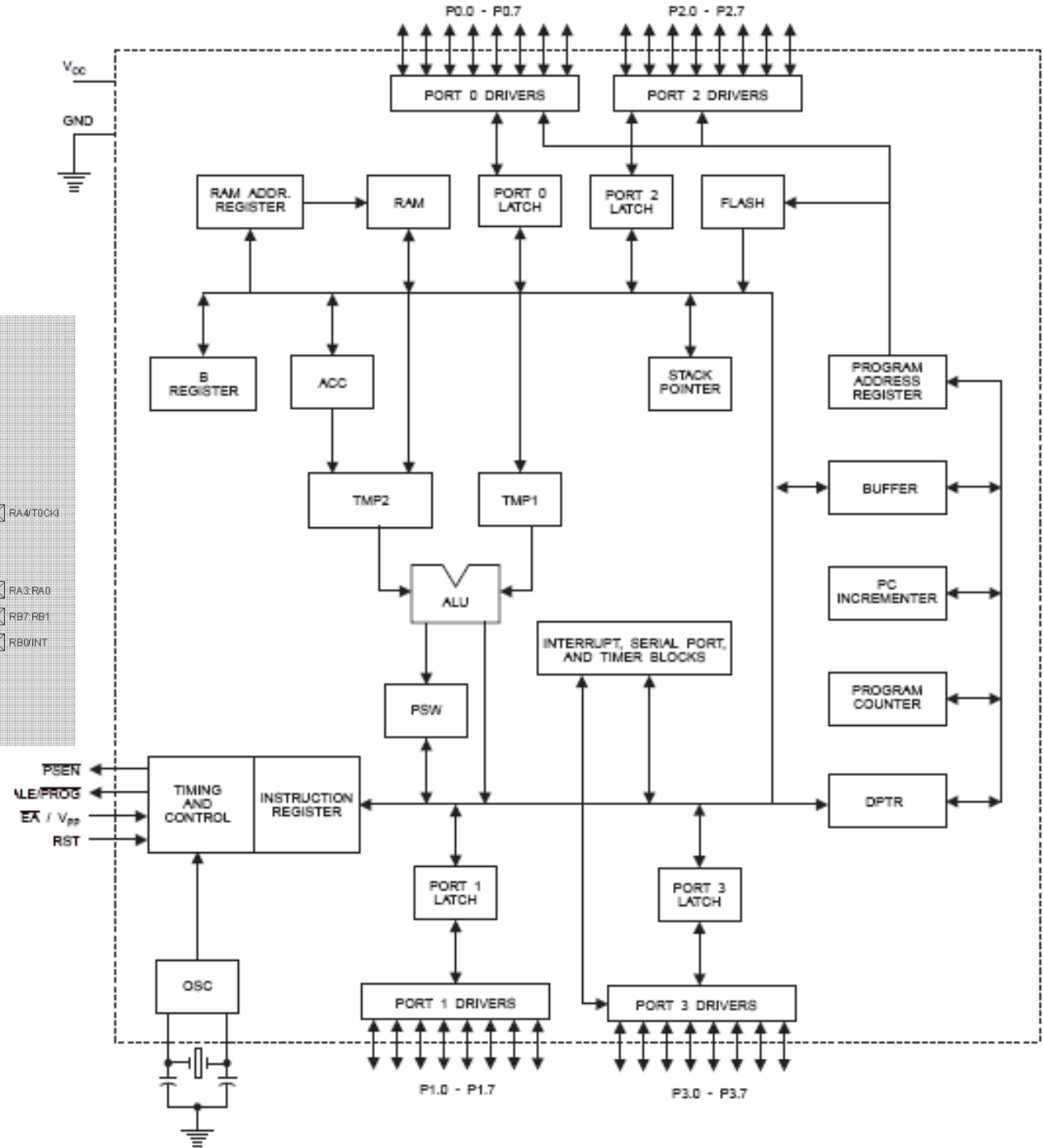
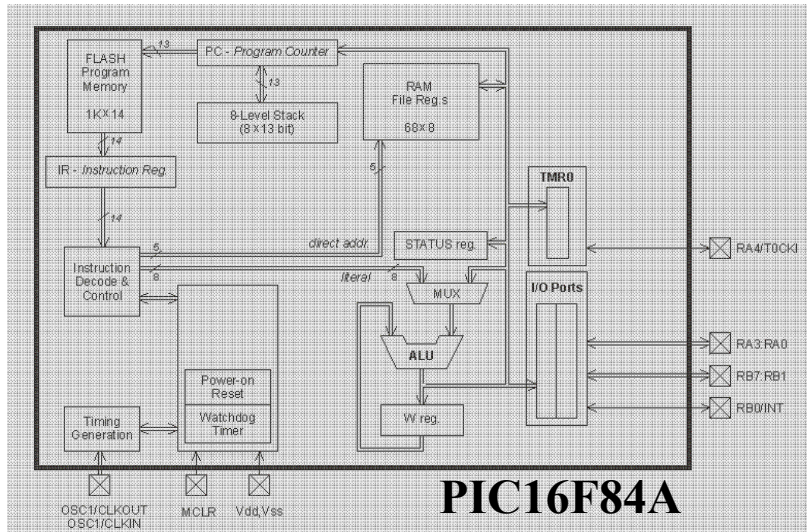


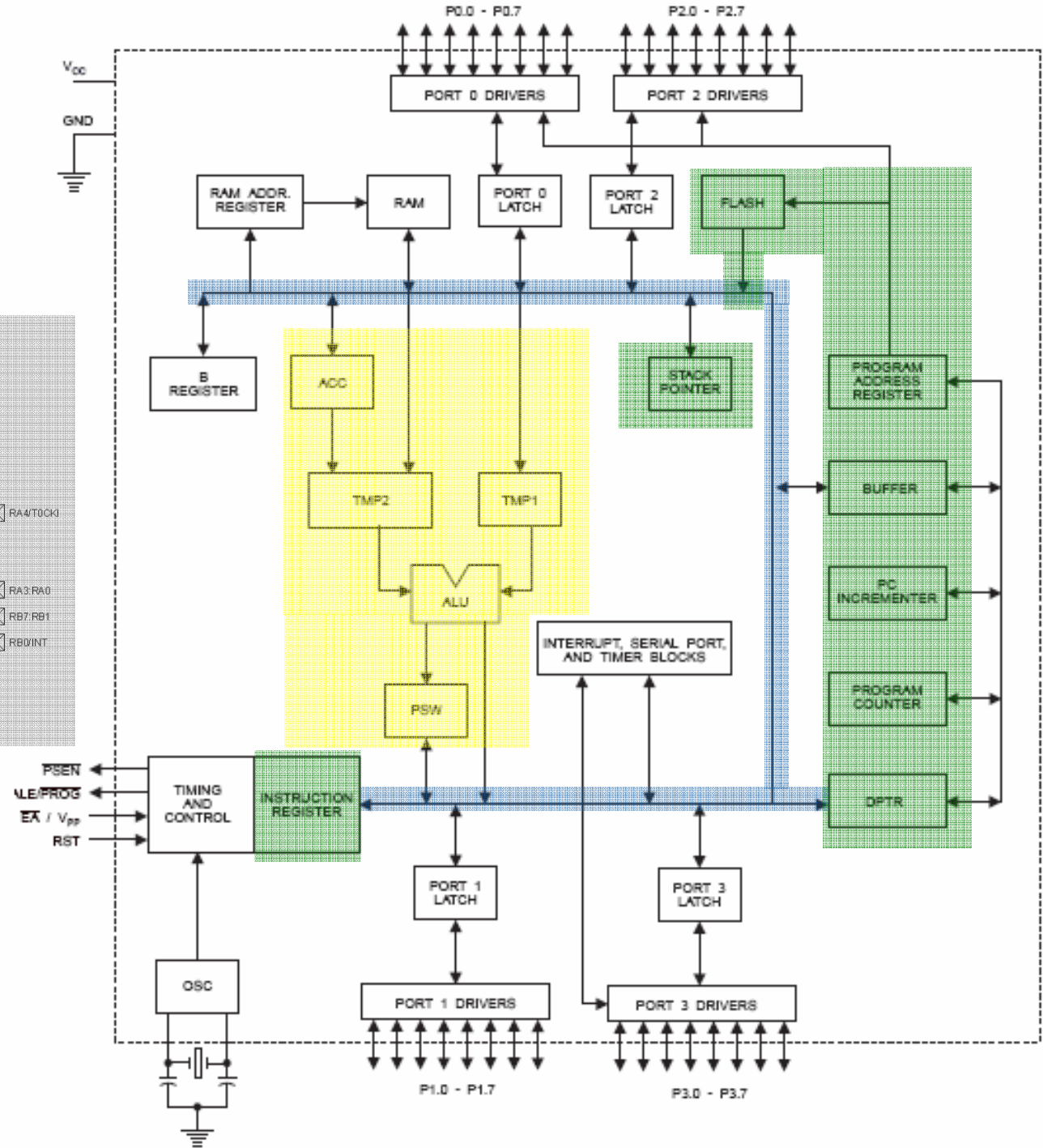
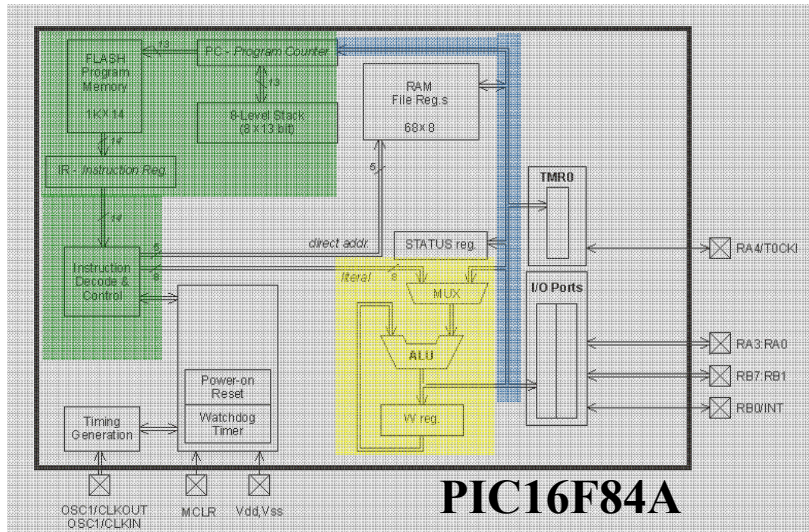
# Szybki przegląd wybranych architektur



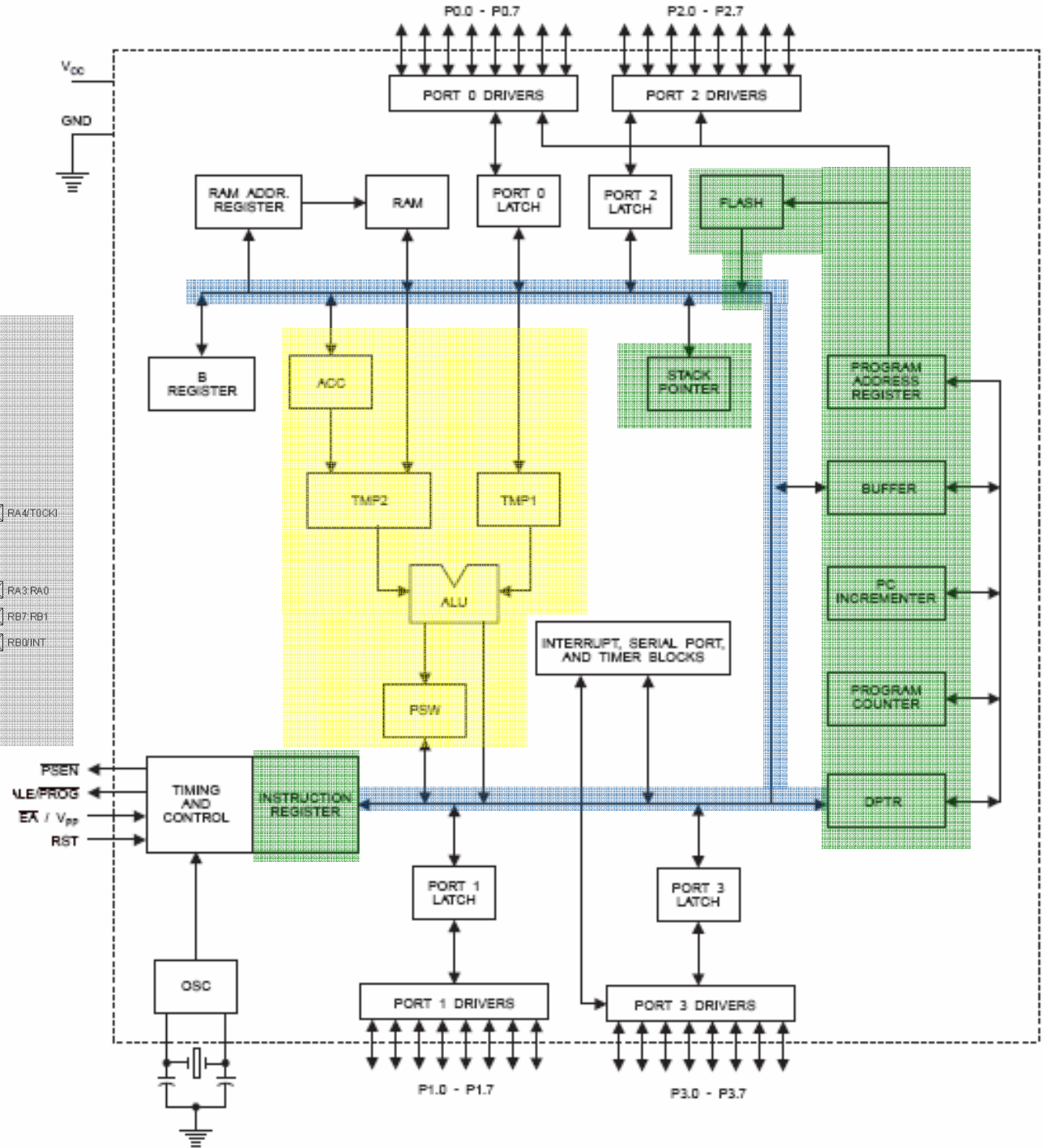
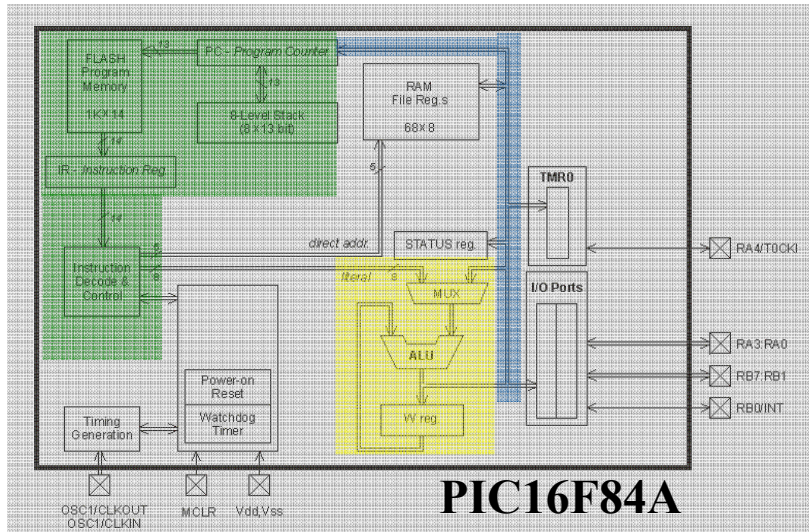
# Architektura nr. 0



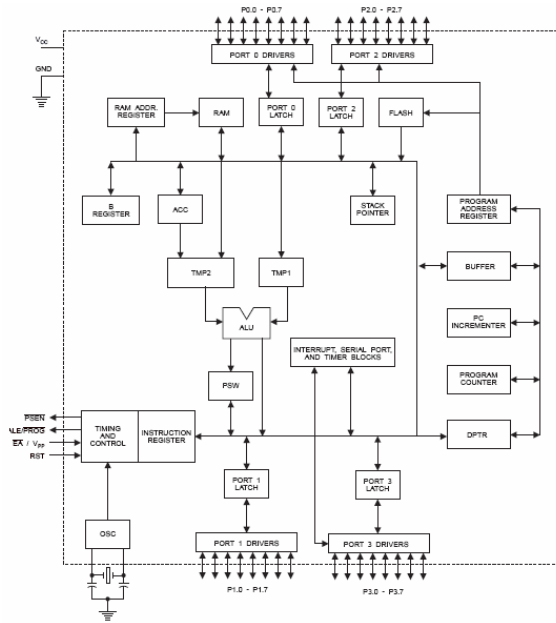
# Architektura nr. 0



# Atmel AT89C51



# Atmel AT89C51



Mnemonic	Description	Byte	Cycle
MOV A,Rn	Move register to accumulator	1	1
MOV A,direct <sup>*)</sup>	Move direct byte to accumulator	2	1
MOV A,@Ri	Move indirect RAM to accumulator	1	1
MOV A,#data	Move immediate data to accumulator	2	1
MOV Rn,A	Move accumulator to register	1	1
MOV Rn,direct	Move direct byte to register	2	2
MOV Rn,#data	Move immediate data to register	2	1
MOV direct,A	Move accumulator to direct byte	2	1
MOV direct,Rn	Move register to direct byte	2	2
MOV direct,direct	Move direct byte to direct byte	3	2
MOV direct,@Ri	Move indirect RAM to direct byte	2	2
MOV direct,#data	Move immediate data to direct byte	3	2
MOV @Ri,A	Move accumulator to indirect RAM	1	1
MOV @Ri,direct	Move direct byte to indirect RAM	2	2
MOV @Ri,#data	Move immediate data to indirect RAM	2	1
MOV DPTR,#data16	Load data pointer with a 16-bit constant	3	2
MOVC A,@A + DPTR	Move code byte relative to DPTR to accumulator	1	2
MOVC A,@A + PC	Move code byte relative to PC to accumulator	1	2
MOVX A,@Ri	Move external RAM (8-bit addr.) to A	1	2
MOVX A,@DPTR	Move external RAM (16-bit addr.) to A	1	2
MOVX @Ri,A	Move A to external RAM (8-bit addr.)	1	2
MOVX @DPTR,A	Move A to external RAM (16-bit addr.)	1	2

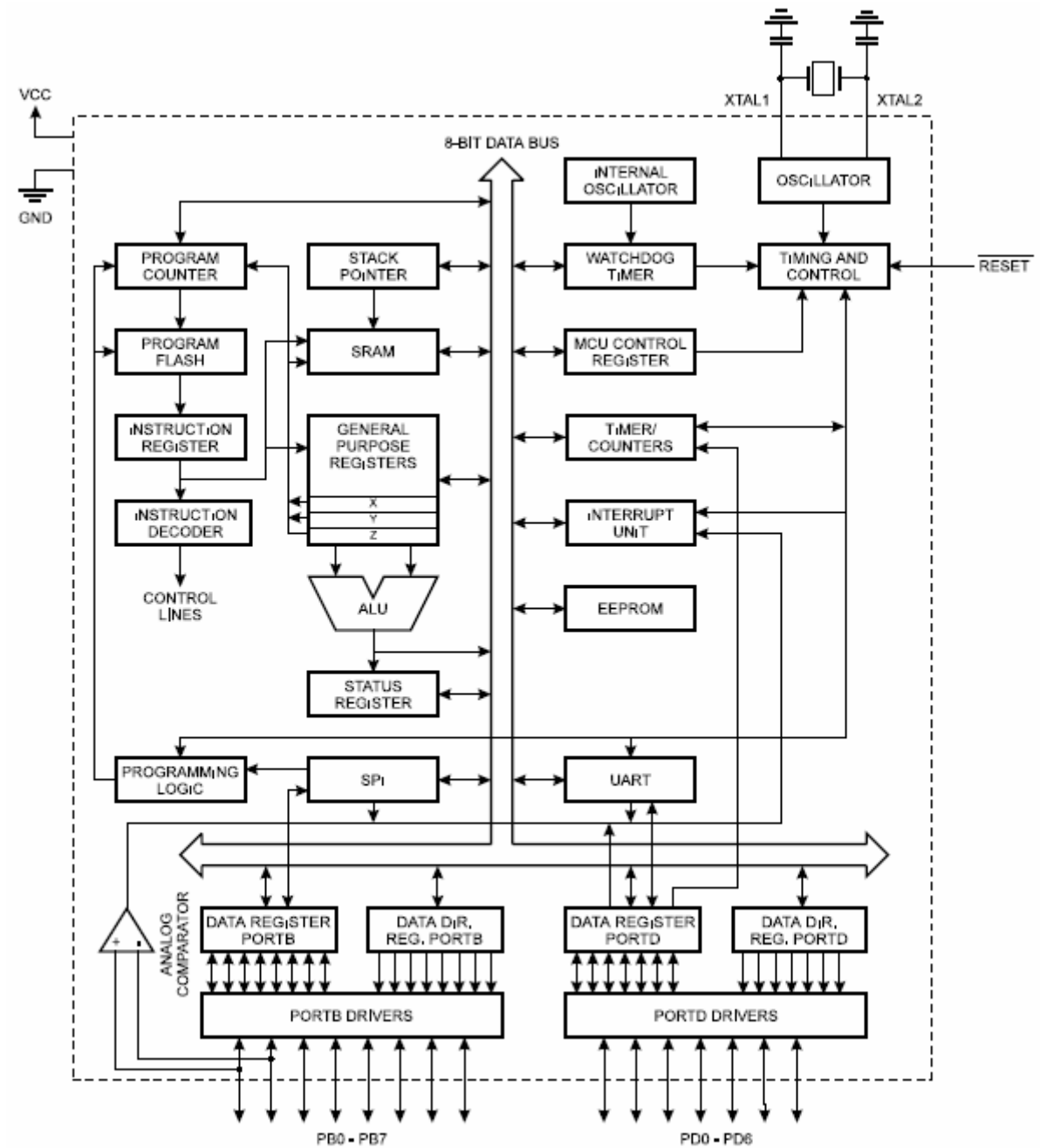
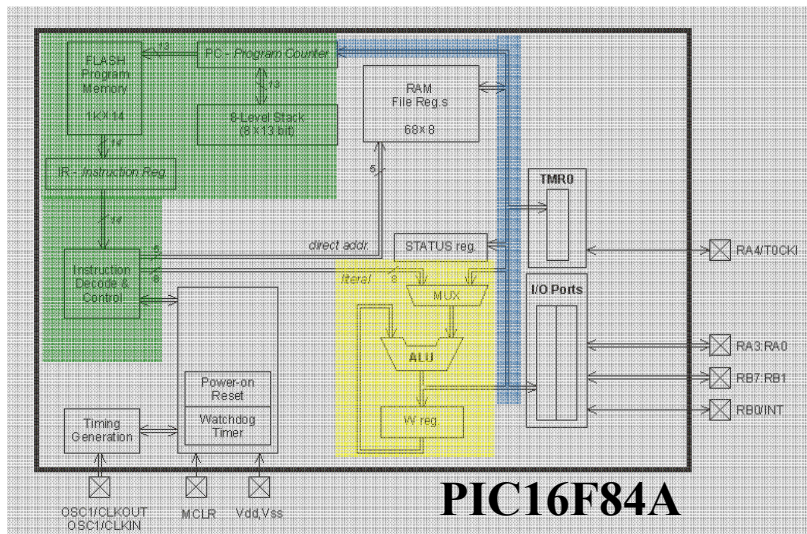
**Notes on Data Addressing Modes**

- Rn - Working register R0-R7
- direct - 128 internal RAM locations, any I/O port, control or status register
- @Ri - Indirect internal or external RAM location addressed by register R0 or R1
- #data - 8-bit constant included in instruction
- #data 16 - 16-bit constant included as bytes 2 and 3 of instruction
- bit - 128 software flags, any bitaddressable I/O pin, control or status bit
- A - Accumulator

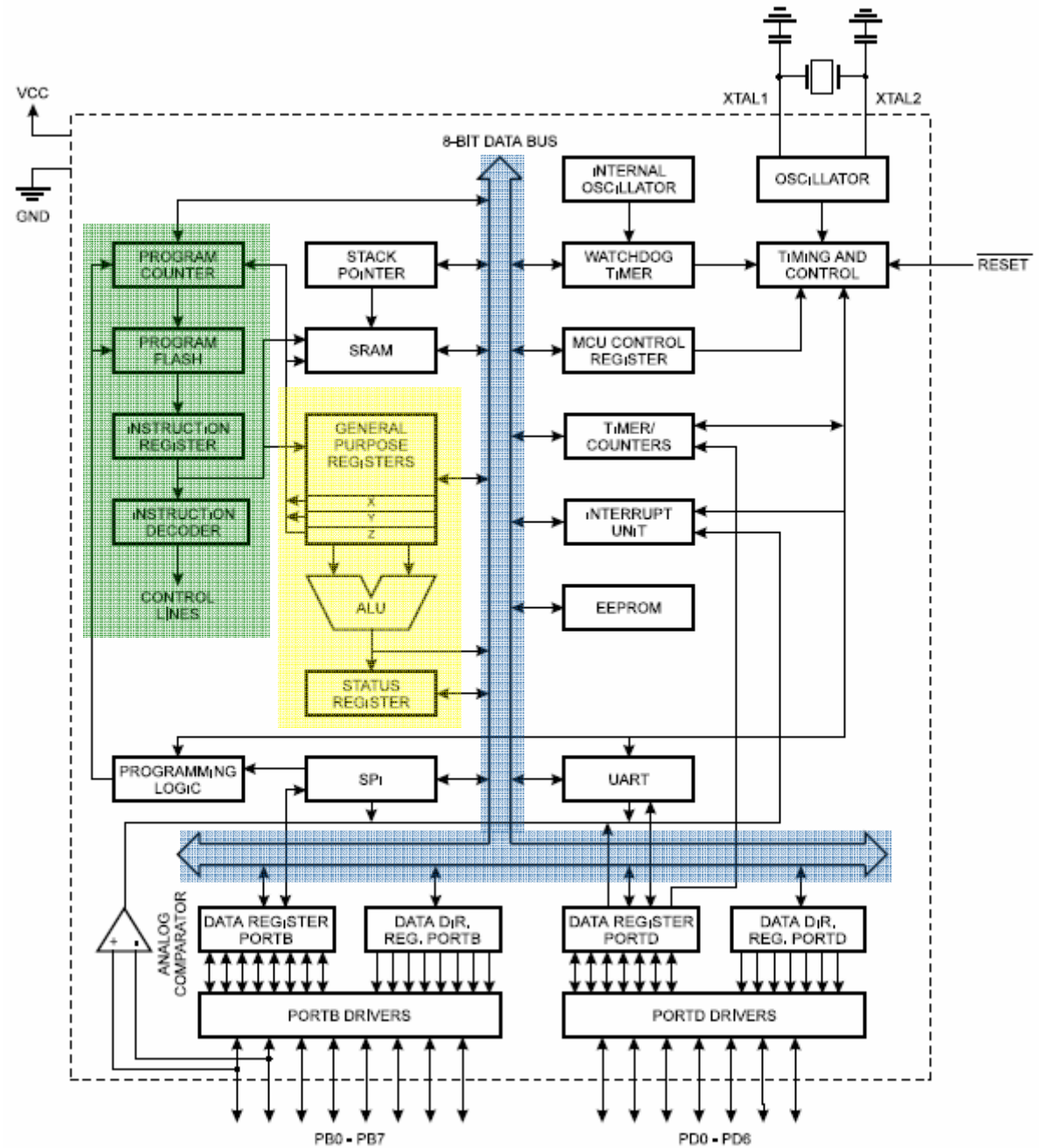
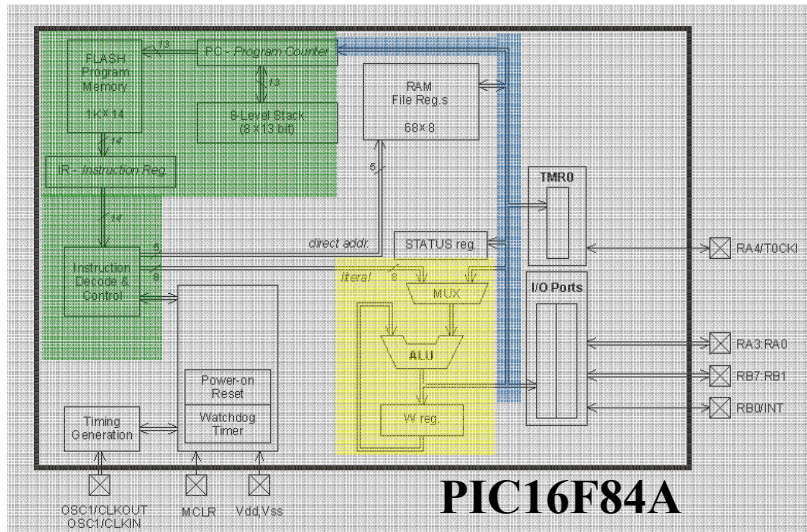
...



# Architektura nr. 1

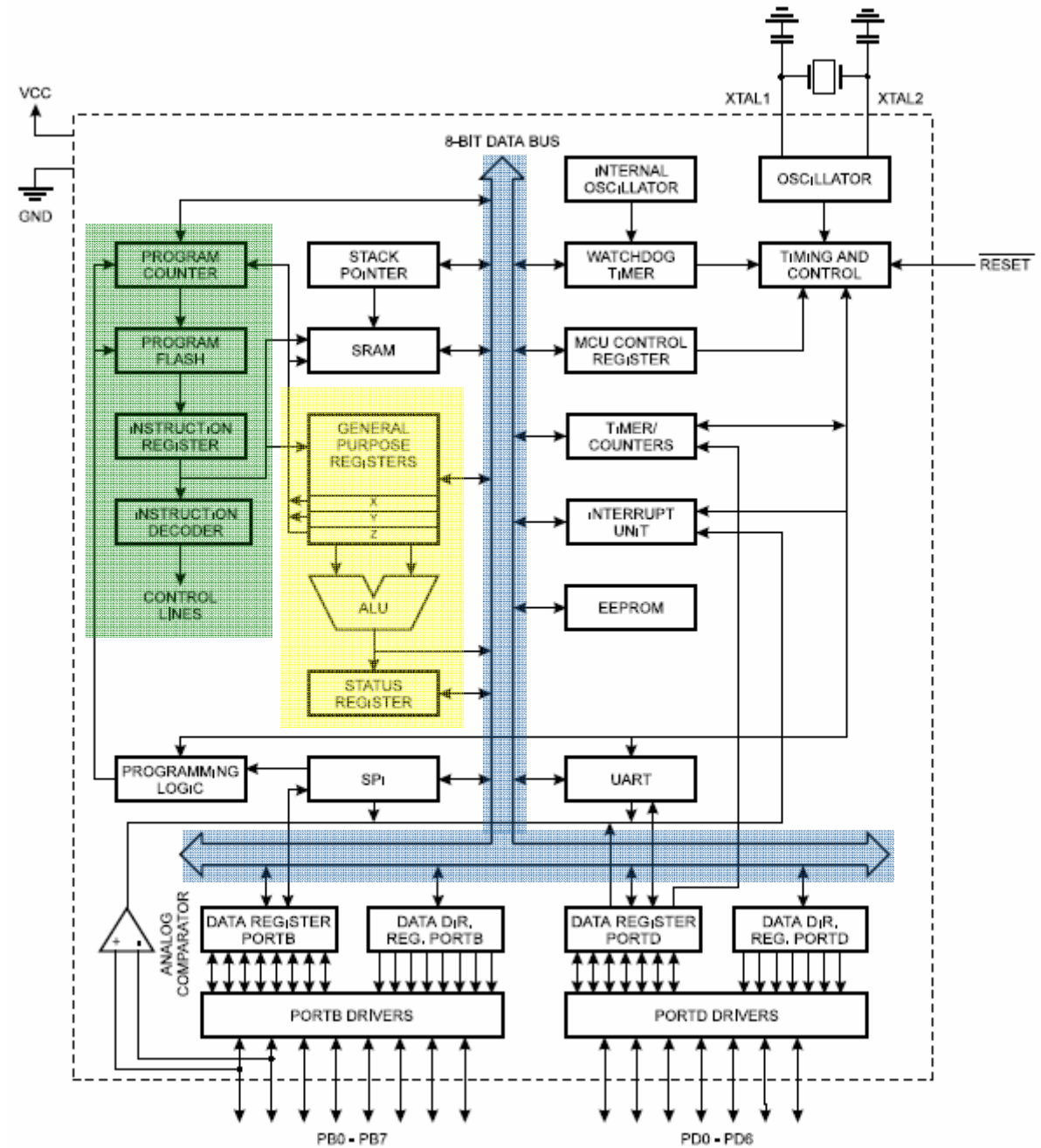
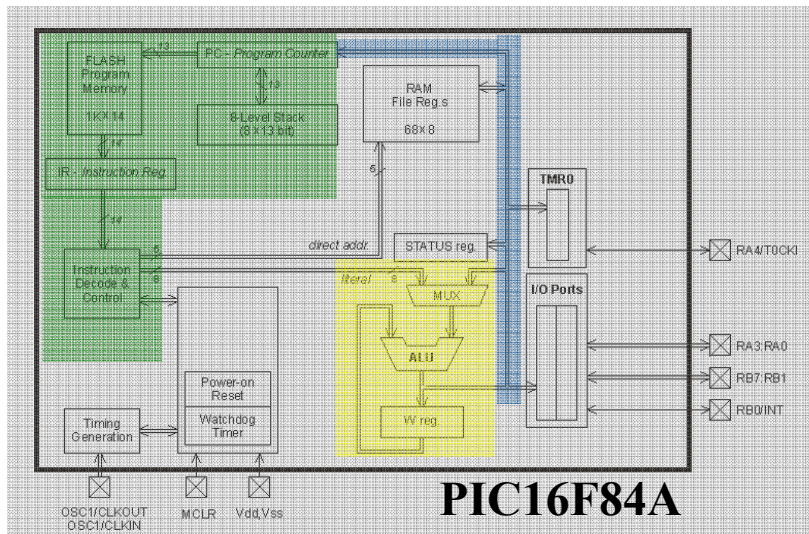


# Architektura nr. 1



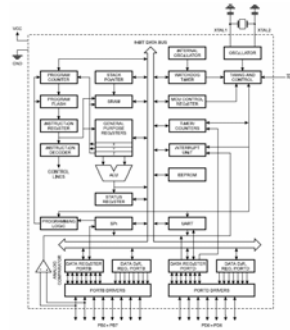
# Atmel AVR

## AT90S2313



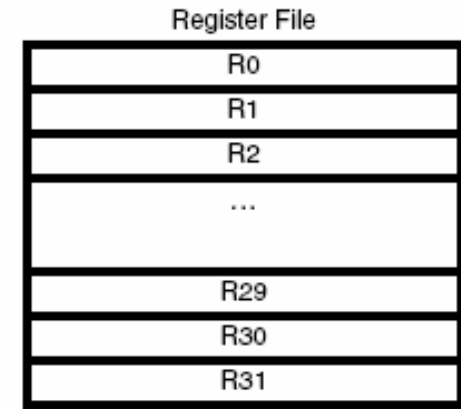
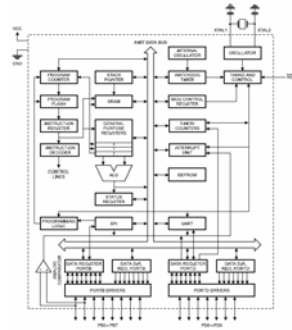


# Atmel AVR AT90S2313

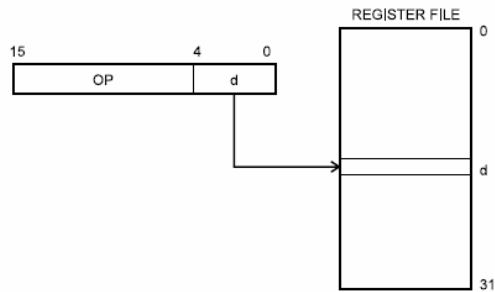


Mnemonic	Operands	Description	Operation	Flags	# Clocks
<b>DATA TRANSFER INSTRUCTIONS</b>					
MOV	Rd, Rr	Move between Registers	$Rd \leftarrow Rr$	None	1
LDI	Rd, K	Load Immediate	$Rd \leftarrow K$	None	1
LD	Rd, X	Load Indirect	$Rd \leftarrow (X)$	None	2
LD	Rd, X+	Load Indirect and Post-Inc.	$Rd \leftarrow (X), X \leftarrow X + 1$	None	2
LD	Rd, -X	Load Indirect and Pre-Dec.	$X \leftarrow X - 1, Rd \leftarrow (X)$	None	2
LD	Rd, Y	Load Indirect	$Rd \leftarrow (Y)$	None	2
LD	Rd, Y+	Load Indirect and Post-Inc.	$Rd \leftarrow (Y), Y \leftarrow Y + 1$	None	2
LD	Rd, -Y	Load Indirect and Pre-Dec.	$Y \leftarrow Y - 1, Rd \leftarrow (Y)$	None	2
LDD	Rd, Y+q	Load Indirect with Displacement	$Rd \leftarrow (Y + q)$	None	2
LD	Rd, Z	Load Indirect	$Rd \leftarrow (Z)$	None	2
LD	Rd, Z+	Load Indirect and Post-Inc.	$Rd \leftarrow (Z), Z \leftarrow Z + 1$	None	2
LD	Rd, -Z	Load Indirect and Pre-Dec.	$Z \leftarrow Z - 1, Rd \leftarrow (Z)$	None	2
LDD	Rd, Z+q	Load Indirect with Displacement	$Rd \leftarrow (Z + q)$	None	2
LDS	Rd, k	Load Direct from SRAM	$Rd \leftarrow (k)$	None	2
ST	X, Rr	Store Indirect	$(X) \leftarrow Rr$	None	2
ST	X+, Rr	Store Indirect and Post-Inc.	$(X) \leftarrow Rr, X \leftarrow X + 1$	None	2
ST	-X, Rr	Store Indirect and Pre-Dec.	$X \leftarrow X - 1, (X) \leftarrow Rr$	None	2
ST	Y, Rr	Store Indirect	$(Y) \leftarrow Rr$	None	2
ST	Y+, Rr	Store Indirect and Post-Inc.	$(Y) \leftarrow Rr, Y \leftarrow Y + 1$	None	2
ST	-Y, Rr	Store Indirect and Pre-Dec.	$Y \leftarrow Y - 1, (Y) \leftarrow Rr$	None	2
STD	Y+q, Rr	Store Indirect with Displacement	$(Y + q) \leftarrow Rr$	None	2
ST	Z, Rr	Store Indirect	$(Z) \leftarrow Rr$	None	2
ST	Z+, Rr	Store Indirect and Post-Inc.	$(Z) \leftarrow Rr, Z \leftarrow Z + 1$	None	2
ST	-Z, Rr	Store Indirect and Pre-Dec.	$Z \leftarrow Z - 1, (Z) \leftarrow Rr$	None	2
STD	Z+q, Rr	Store Indirect with Displacement	$(Z + q) \leftarrow Rr$	None	2
STS	k, Rr	Store Direct to SRAM	$(k) \leftarrow Rr$	None	2
LPM		Load Program Memory	$R0 \leftarrow (Z)$	None	3
IN	Rd, P	In Port	$Rd \leftarrow P$	None	1
OUT	P, Rr	Out Port	$P \leftarrow Rr$	None	1
PUSH	Rr	Push Register on Stack	$STACK \leftarrow Rr$	None	2
POP	Rd	Pop Register from Stack	$Rd \leftarrow STACK$	None	2

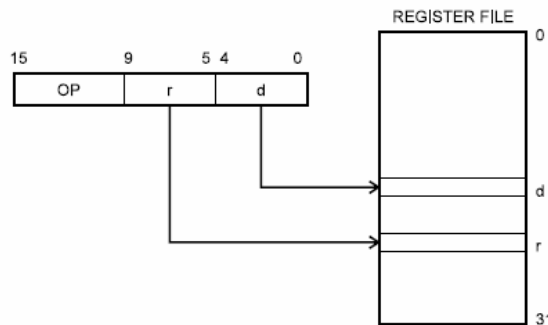
# Atmel AVR AT90S2313



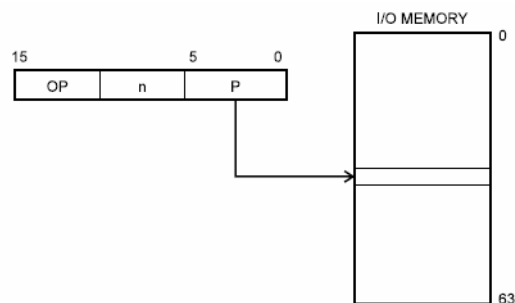
Register Direct, Single Register Rd



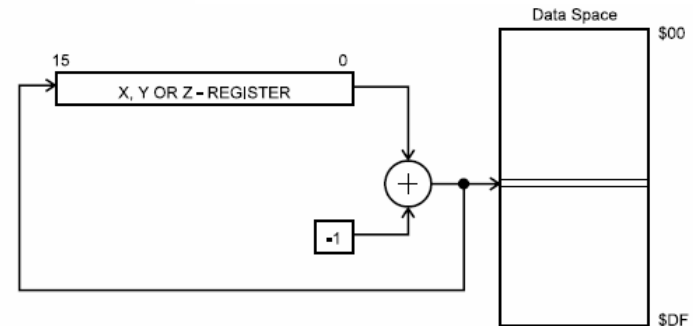
Register Direct, Two Registers Rd and Rr



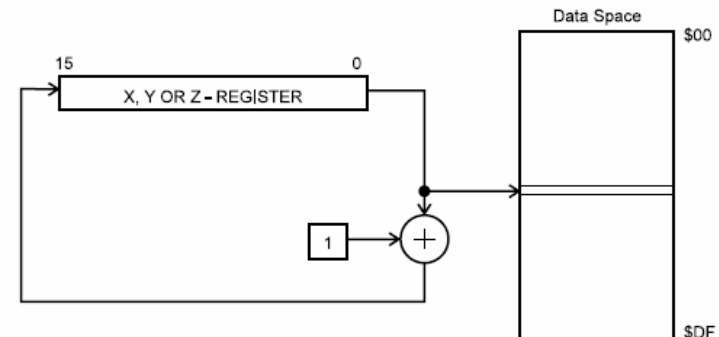
I/O Direct



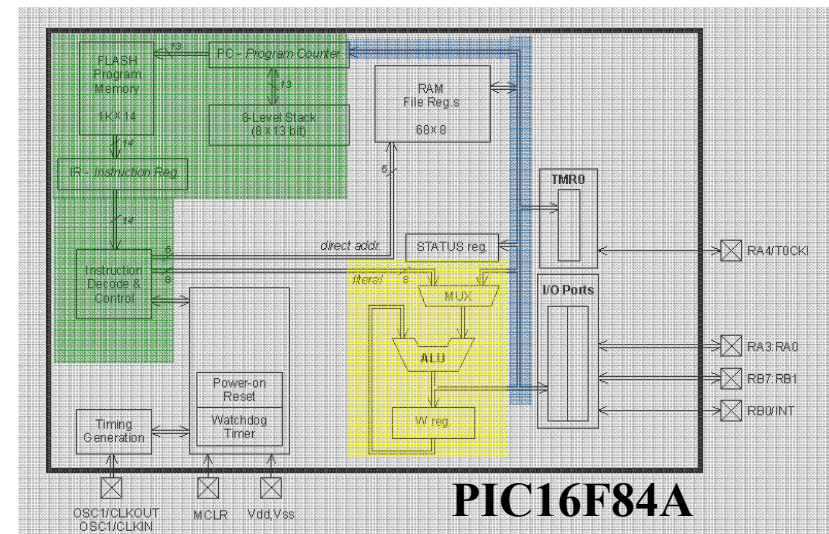
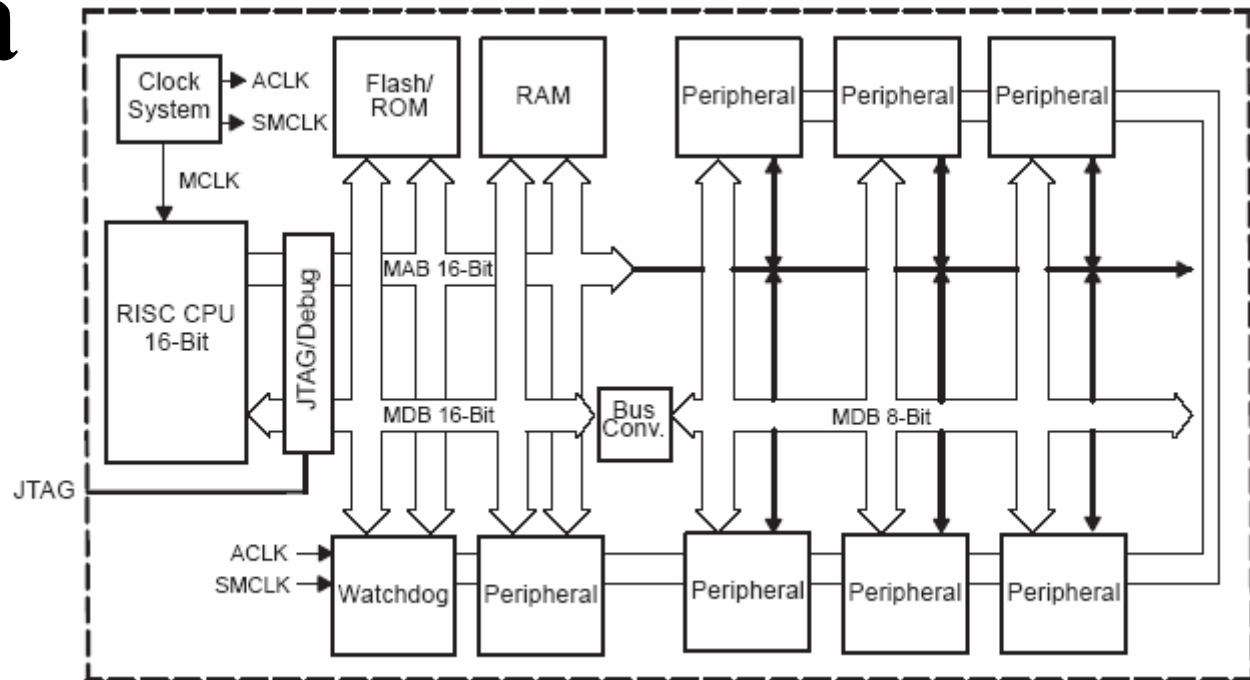
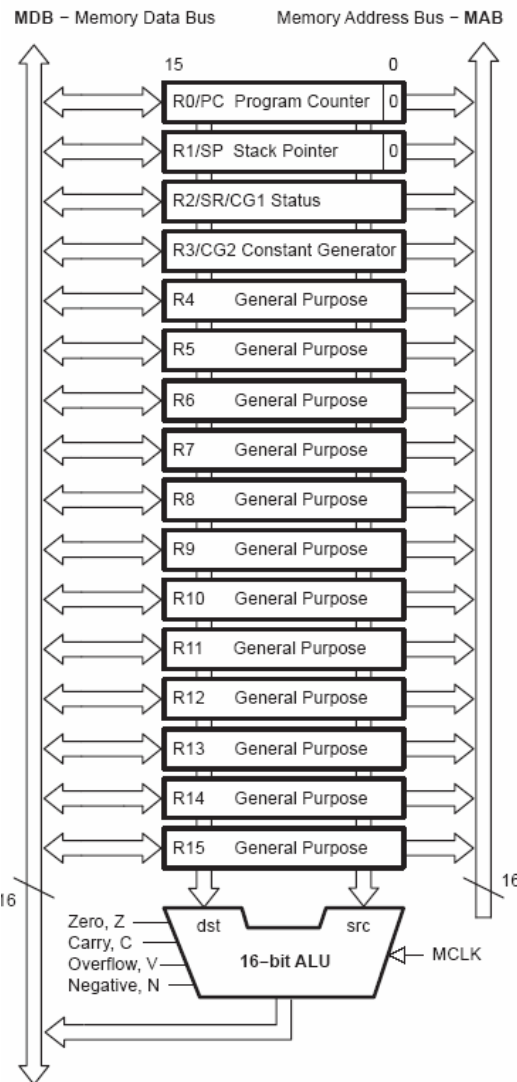
Data Indirect with Pre-decrement



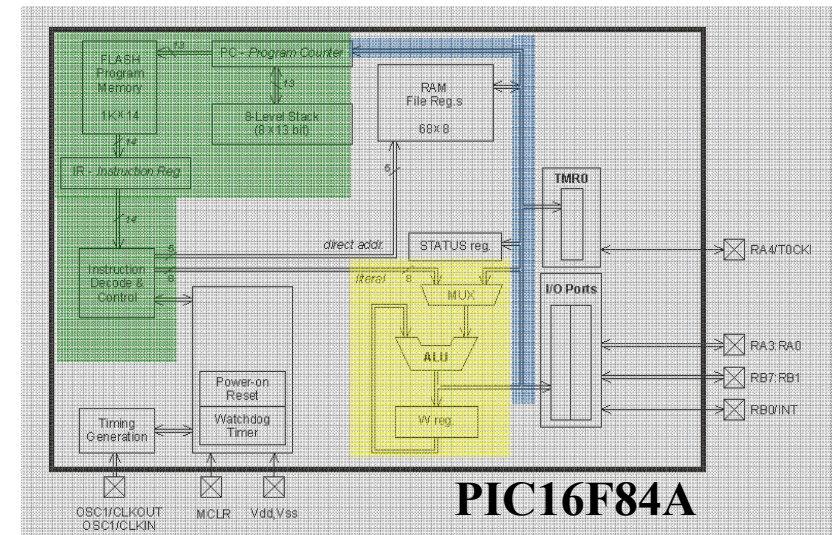
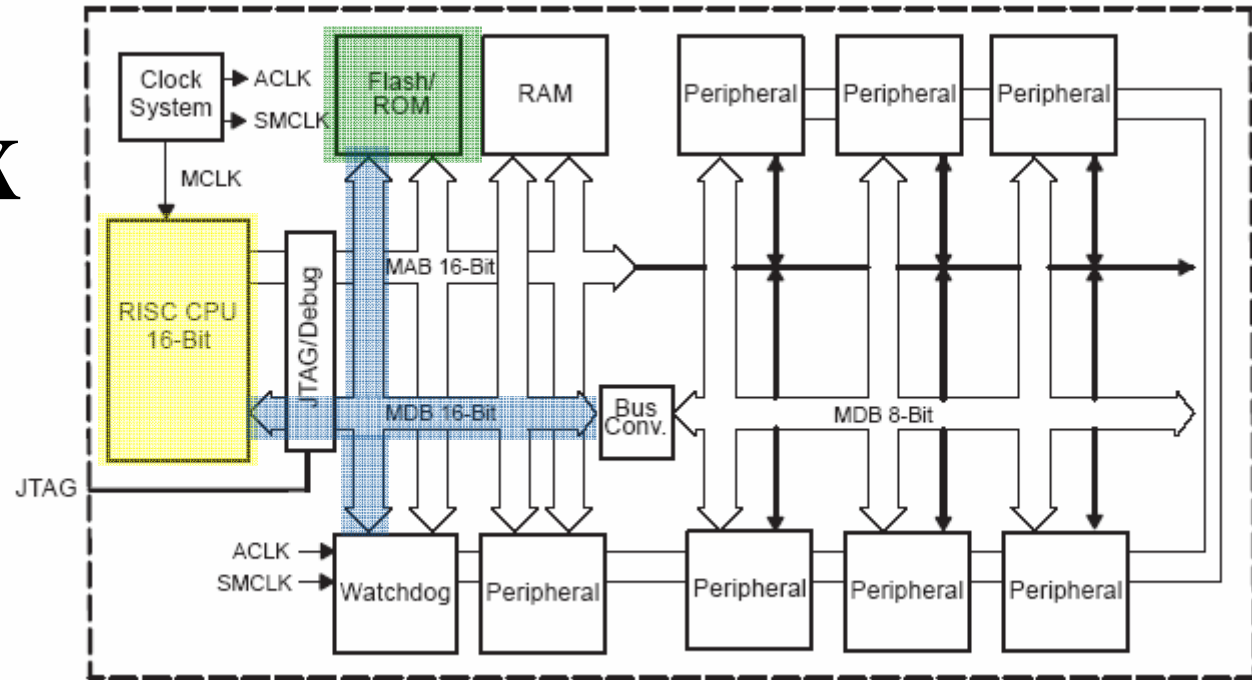
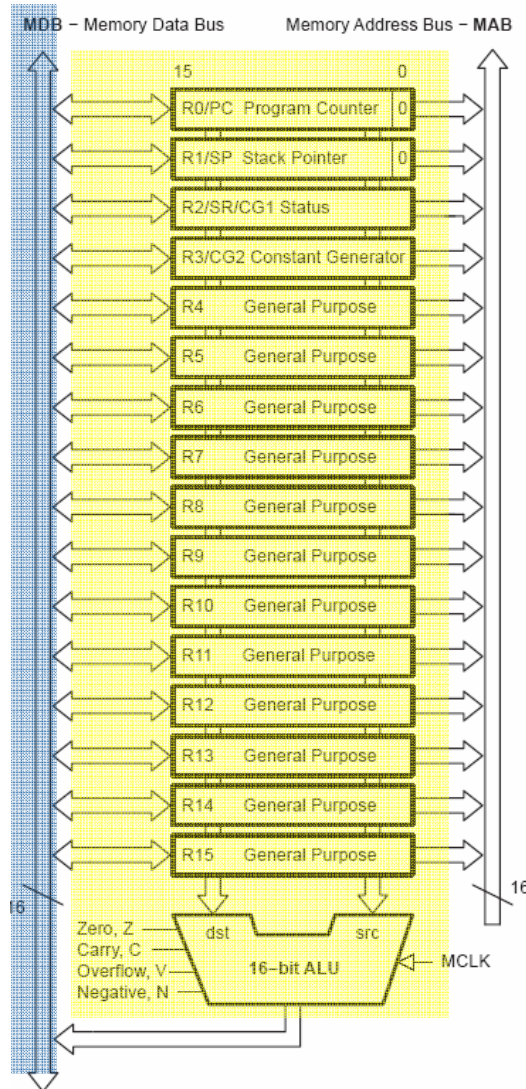
Data Indirect with Post-increment



# Architektura nr. 2



# Texas MSP430x1xx



# Texas

## MSP430x1xx

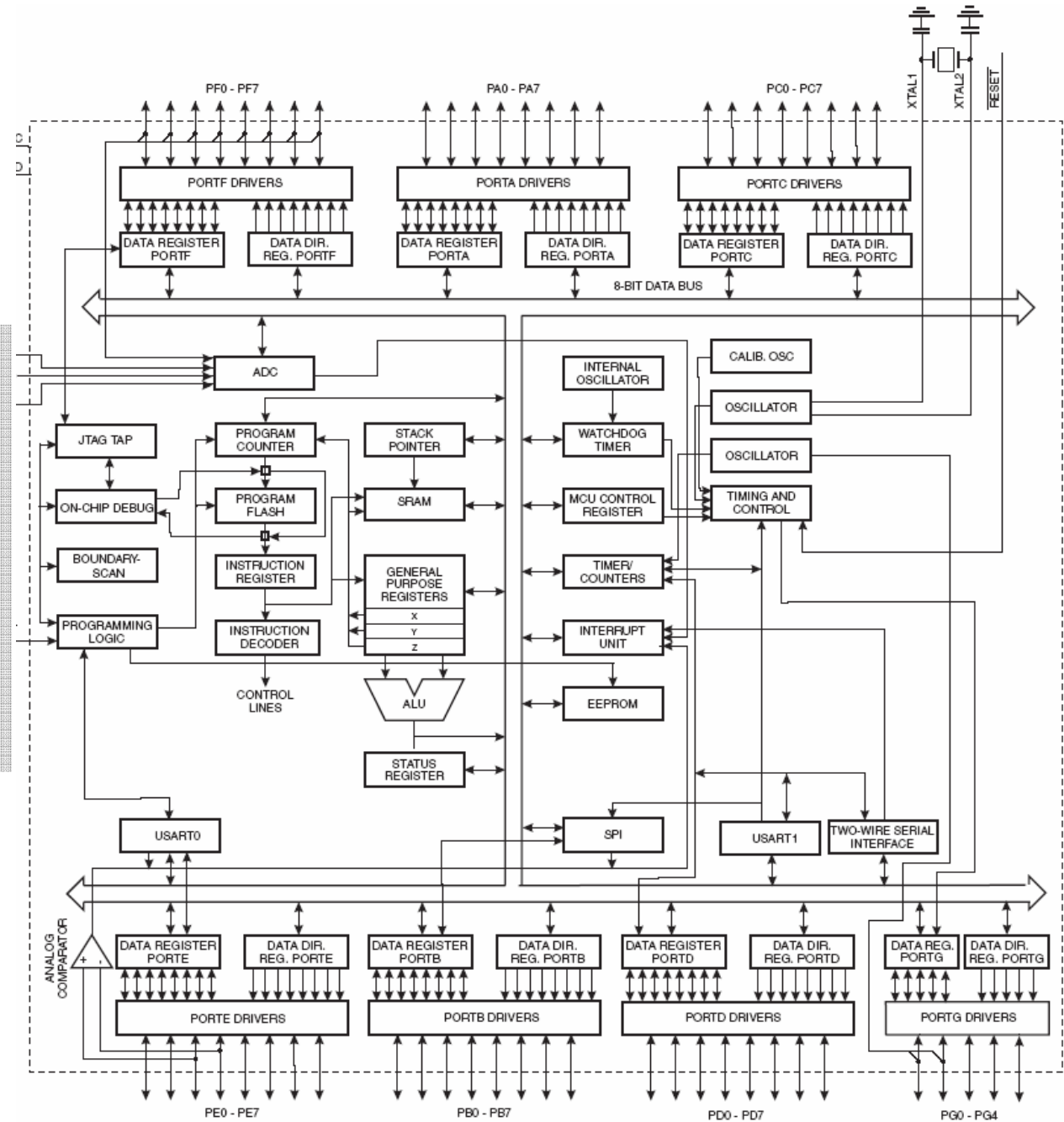
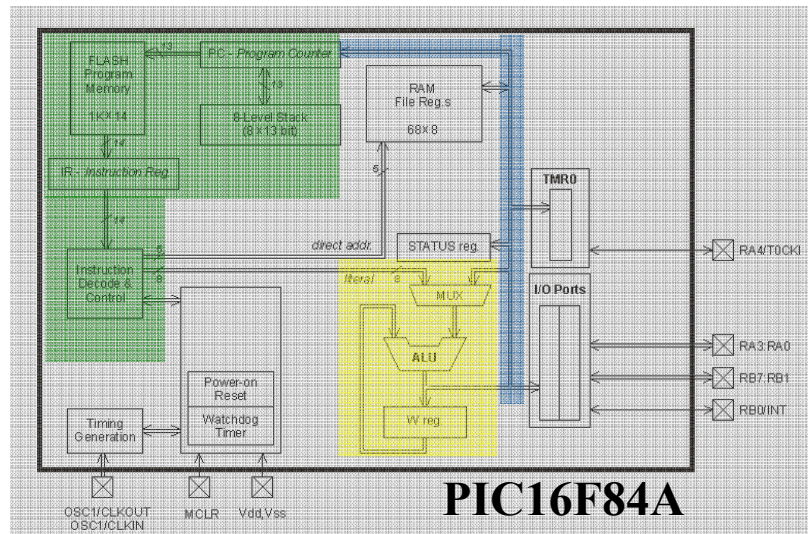
As/Ad	Addressing Mode	Syntax	Description
00/0	Register mode	Rn	Register contents are operand
01/1	Indexed mode	X(Rn)	(Rn + X) points to the operand. X is stored in the next word.
01/1	Symbolic mode	ADDR	(PC + X) points to the operand. X is stored in the next word. Indexed mode X(PC) is used.
01/1	Absolute mode	&ADDR	The word following the instruction contains the absolute address. X is stored in the next word. Indexed mode X(SR) is used.
10/-	Indirect register mode	@Rn	Rn is used as a pointer to the operand.
11/-	Indirect autoincrement	@Rn+	Rn is used as a pointer to the operand. Rn is incremented afterwards by 1 for .B instructions and by 2 for .W instructions.
11/-	Immediate mode	#N	The word following the instruction contains the immediate constant N. Indirect autoincrement mode @PC+ is used.

Mnemonic	S-Reg, D-Reg	Operation	Status Bits			
			V	N	Z	C
MOV (.B)	src, dst	src → dst	-	-	-	-
ADD (.B)	src, dst	src + dst → dst	*	*	*	*
ADDC (.B)	src, dst	src + dst + C → dst	*	*	*	*
SUB (.B)	src, dst	dst + .not.src + 1 → dst	*	*	*	*
SUBC (.B)	src, dst	dst + .not.src + C → dst	*	*	*	*
CMP (.B)	src, dst	dst - src	*	*	*	*
DADD (.B)	src, dst	src + dst + C → dst (decimally)	*	*	*	*
BIT (.B)	src, dst	src .and. dst	0	*	*	*
BIC (.B)	src, dst	.not.src .and. dst → dst	-	-	-	-
BIS (.B)	src, dst	src .or. dst → dst	-	-	-	-
XOR (.B)	src, dst	src .xor. dst → dst	*	*	*	*
AND (.B)	src, dst	src .and. dst → dst	0	*	*	*

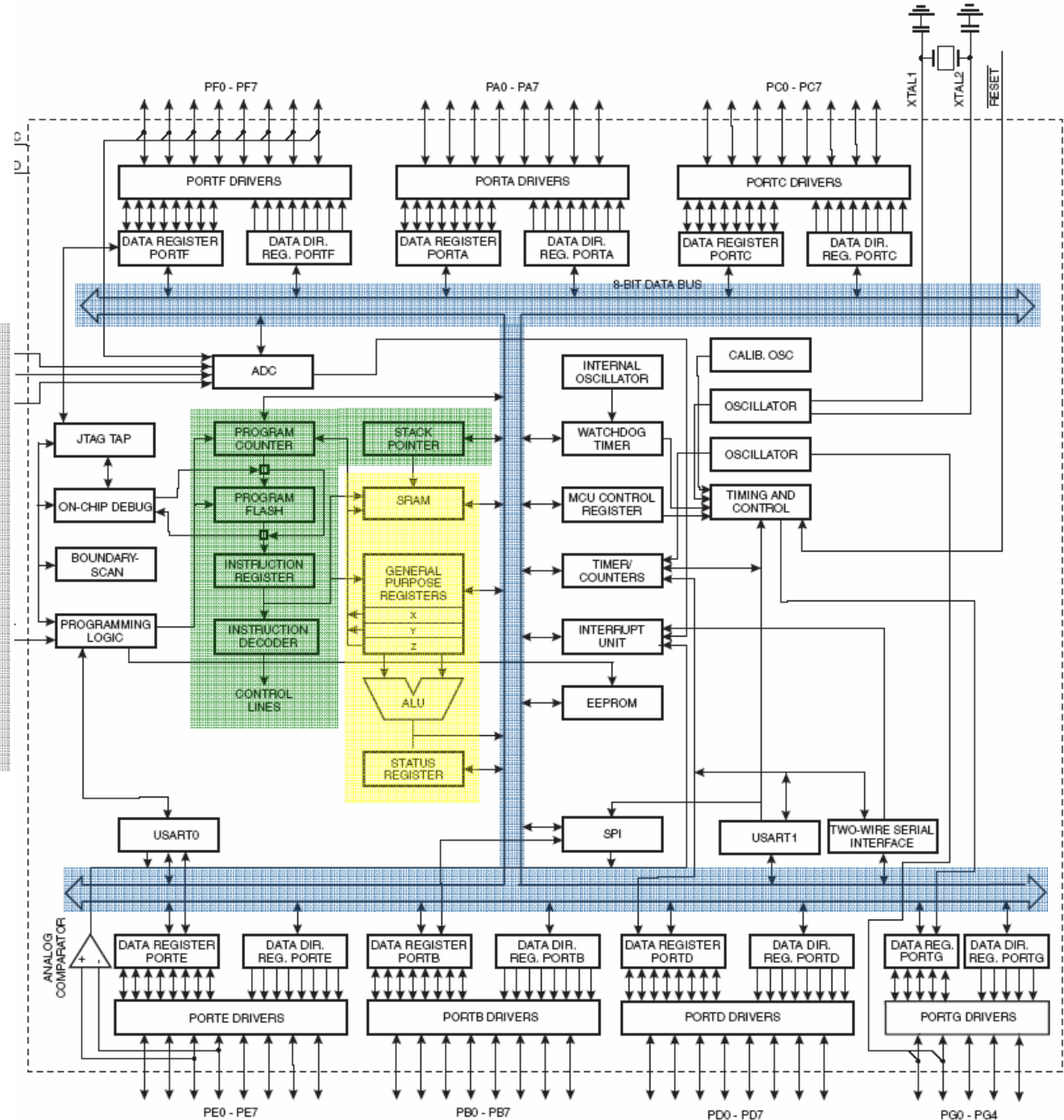
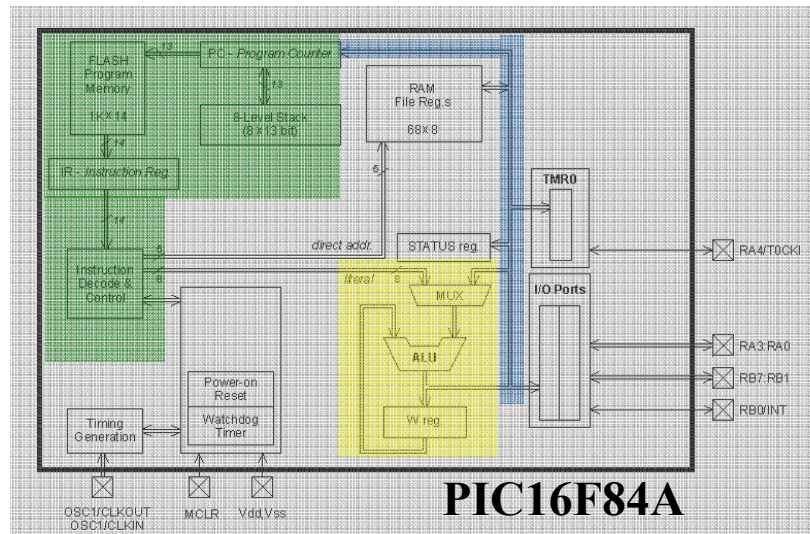
27 instructions and 7 addressing modes.



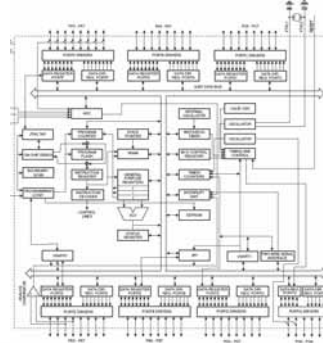
# Architektura nr. 3



# Architektura nr. 3

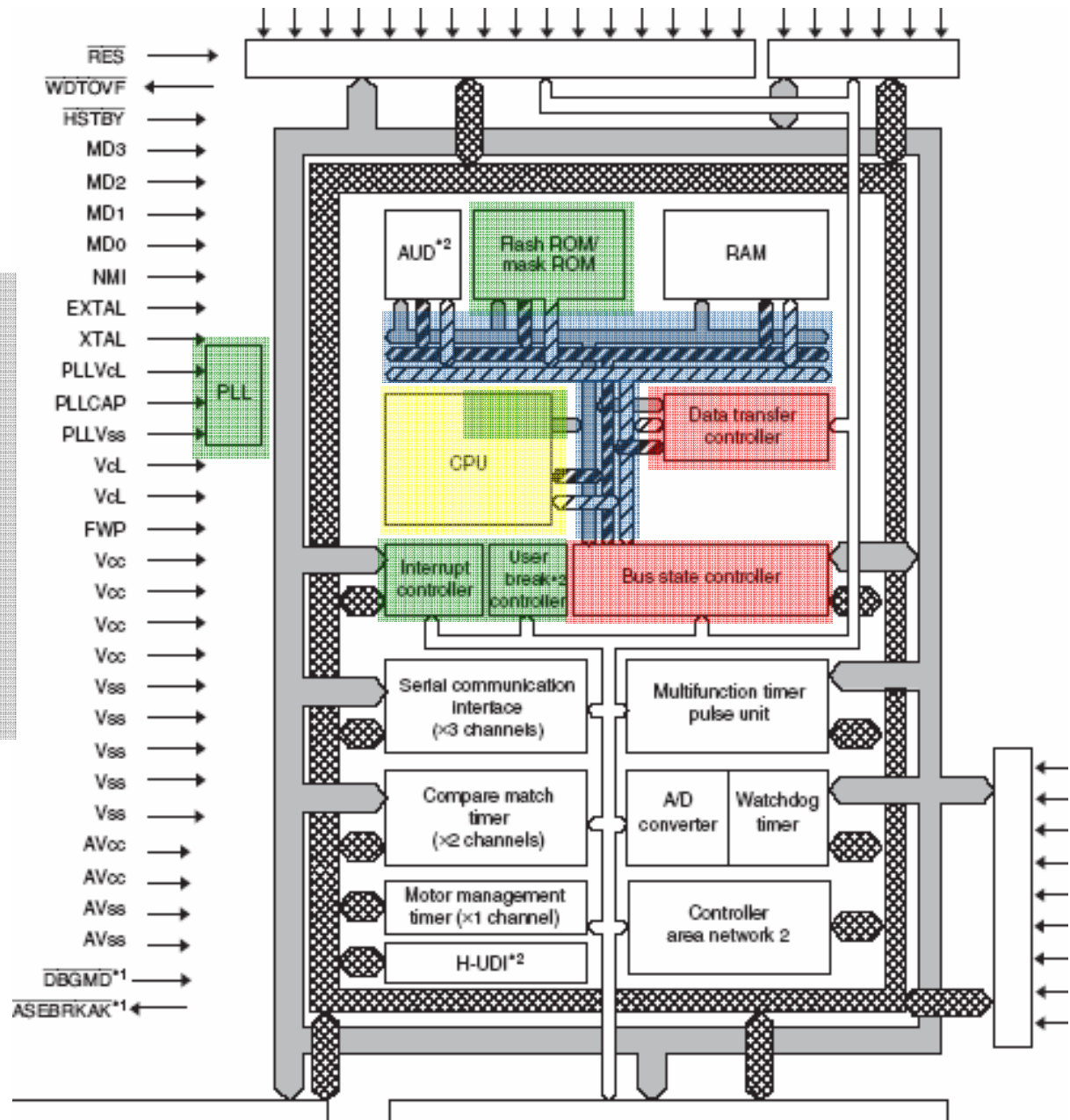
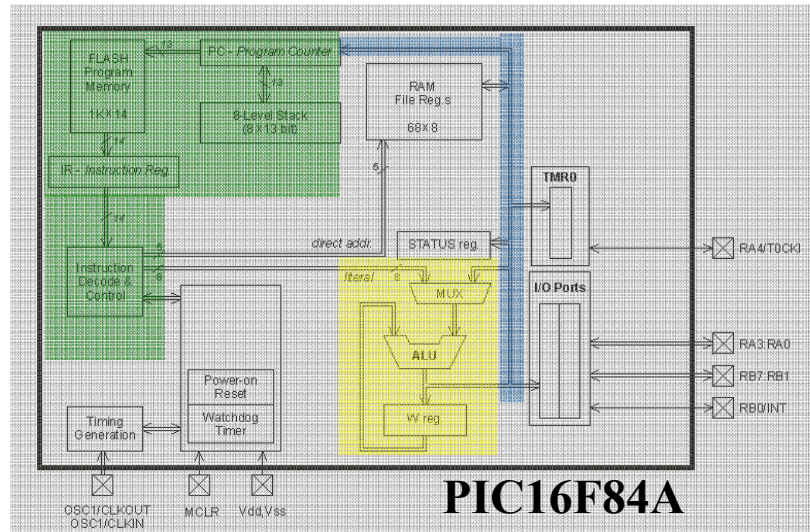


# Atmel AVR ATMega128



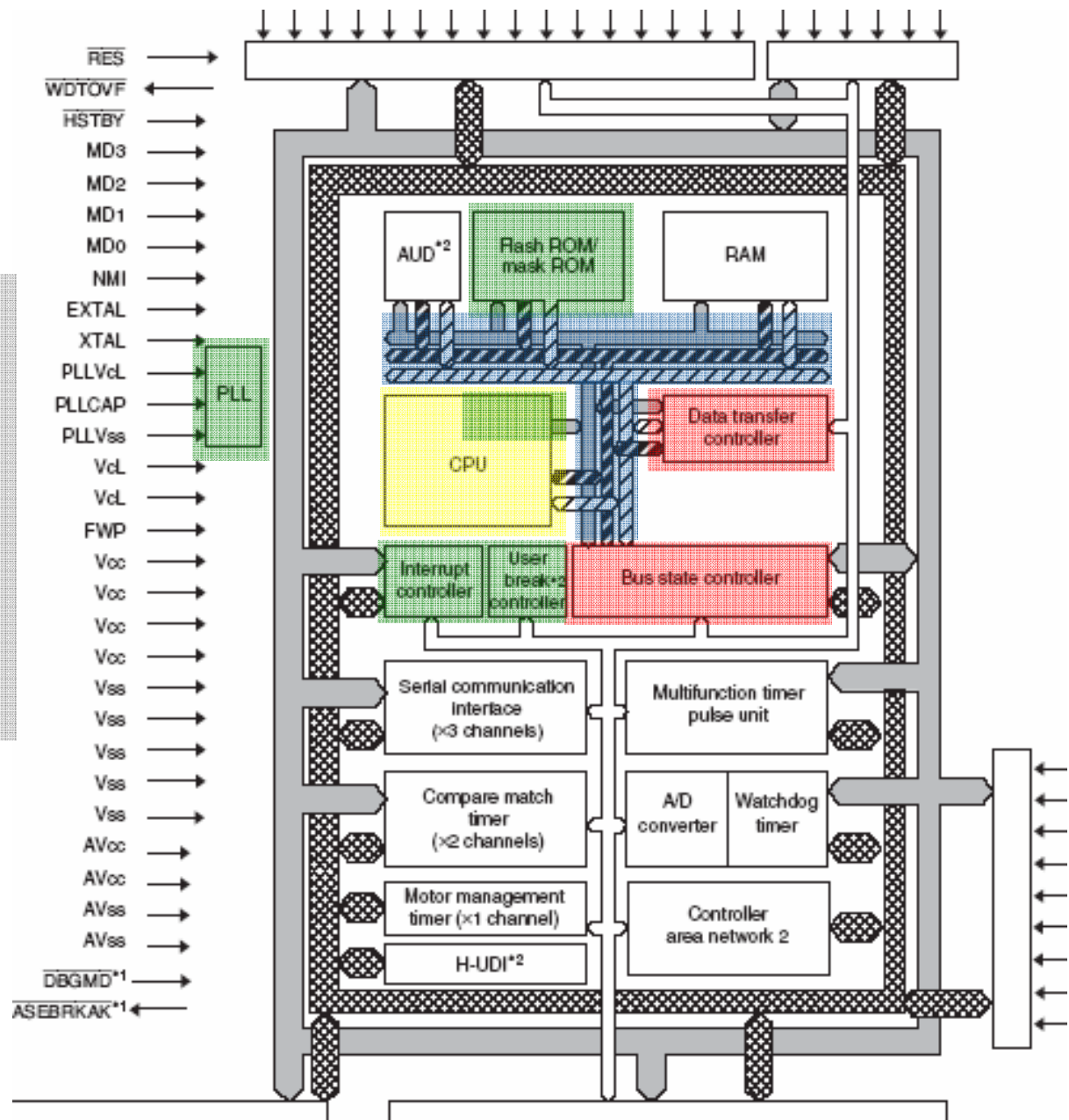
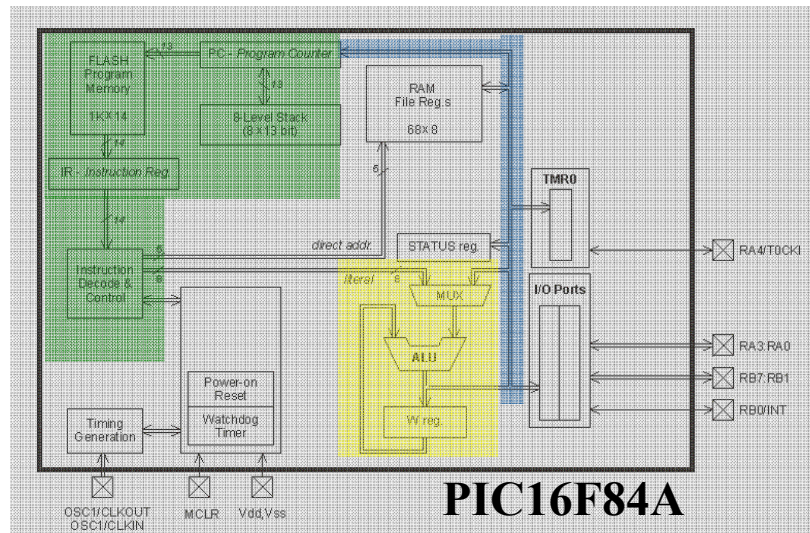
Mnemonics	Operands	Description	Operation	Flags	#Clocks
<b>ARITHMETIC AND LOGIC INSTRUCTIONS</b>					
ADD	Rd, Rr	Add two Registers	$Rd \leftarrow Rd + Rr$	Z,C,N,V,H	1
ADC	Rd, Rr	Add with Carry two Registers	$Rd \leftarrow Rd + Rr + C$	Z,C,N,V,H	1
ADIW	RdI,K	Add Immediate to Word	$RdH:RdL \leftarrow RdH:RdL + K$	Z,C,N,V,S	2
SUB	Rd, Rr	Subtract two Registers	$Rd \leftarrow Rd - Rr$	Z,C,N,V,H	1
SUBI	Rd, K	Subtract Constant from Register	$Rd \leftarrow Rd - K$	Z,C,N,V,H	1
SBC	Rd, Rr	Subtract with Carry two Registers	$Rd \leftarrow Rd - Rr - C$	Z,C,N,V,H	1
SBCI	Rd, K	Subtract with Carry Constant from Reg.	$Rd \leftarrow Rd - K - C$	Z,C,N,V,H	1
SBIW	RdI,K	Subtract Immediate from Word	$RdH:RdL \leftarrow RdH:RdL - K$	Z,C,N,V,S	2
AND	Rd, Rr	Logical AND Registers	$Rd \leftarrow Rd \cdot Rr$	Z,N,V	1
ANDI	Rd, K	Logical AND Register and Constant	$Rd \leftarrow Rd \cdot K$	Z,N,V	1
OR	Rd, Rr	Logical OR Registers	$Rd \leftarrow Rd \vee Rr$	Z,N,V	1
ORI	Rd, K	Logical OR Register and Constant	$Rd \leftarrow Rd \vee K$	Z,N,V	1
EOR	Rd, Rr	Exclusive OR Registers	$Rd \leftarrow Rd \oplus Rr$	Z,N,V	1
COM	Rd	One's Complement	$Rd \leftarrow \$FF - Rd$	Z,C,N,V	1
NEG	Rd	Two's Complement	$Rd \leftarrow \$00 - Rd$	Z,C,N,V,H	1
SBR	Rd,K	Set Bit(s) in Register	$Rd \leftarrow Rd \vee K$	Z,N,V	1
CBR	Rd,K	Clear Bit(s) in Register	$Rd \leftarrow Rd \cdot (\$FF - K)$	Z,N,V	1
INC	Rd	Increment	$Rd \leftarrow Rd + 1$	Z,N,V	1
DEC	Rd	Decrement	$Rd \leftarrow Rd - 1$	Z,N,V	1
TST	Rd	Test for Zero or Minus	$Rd \leftarrow Rd \cdot Rd$	Z,N,V	1
CLR	Rd	Clear Register	$Rd \leftarrow Rd \oplus Rd$	Z,N,V	1
SER	Rd	Set Register	$Rd \leftarrow \$FF$	None	1
MUL	Rd, Rr	Multiply Unsigned	$R1:R0 \leftarrow Rd \times Rr$	Z,C	2
MULS	Rd, Rr	Multiply Signed	$R1:R0 \leftarrow Rd \times Rr$	Z,C	2
MULSU	Rd, Rr	Multiply Signed with Unsigned	$R1:R0 \leftarrow Rd \times Rr$	Z,C	2
FMUL	Rd, Rr	Fractional Multiply Unsigned	$R1:R0 \leftarrow (Rd \times Rr) \lll 1$	Z,C	2
FMULS	Rd, Rr	Fractional Multiply Signed	$R1:R0 \leftarrow (Rd \times Rr) \lll 1$	Z,C	2
FMULSU	Rd, Rr	Fractional Multiply Signed with Unsigned	$R1:R0 \leftarrow (Rd \times Rr) \lll 1$	Z,C	2

# Architektura nr. 4



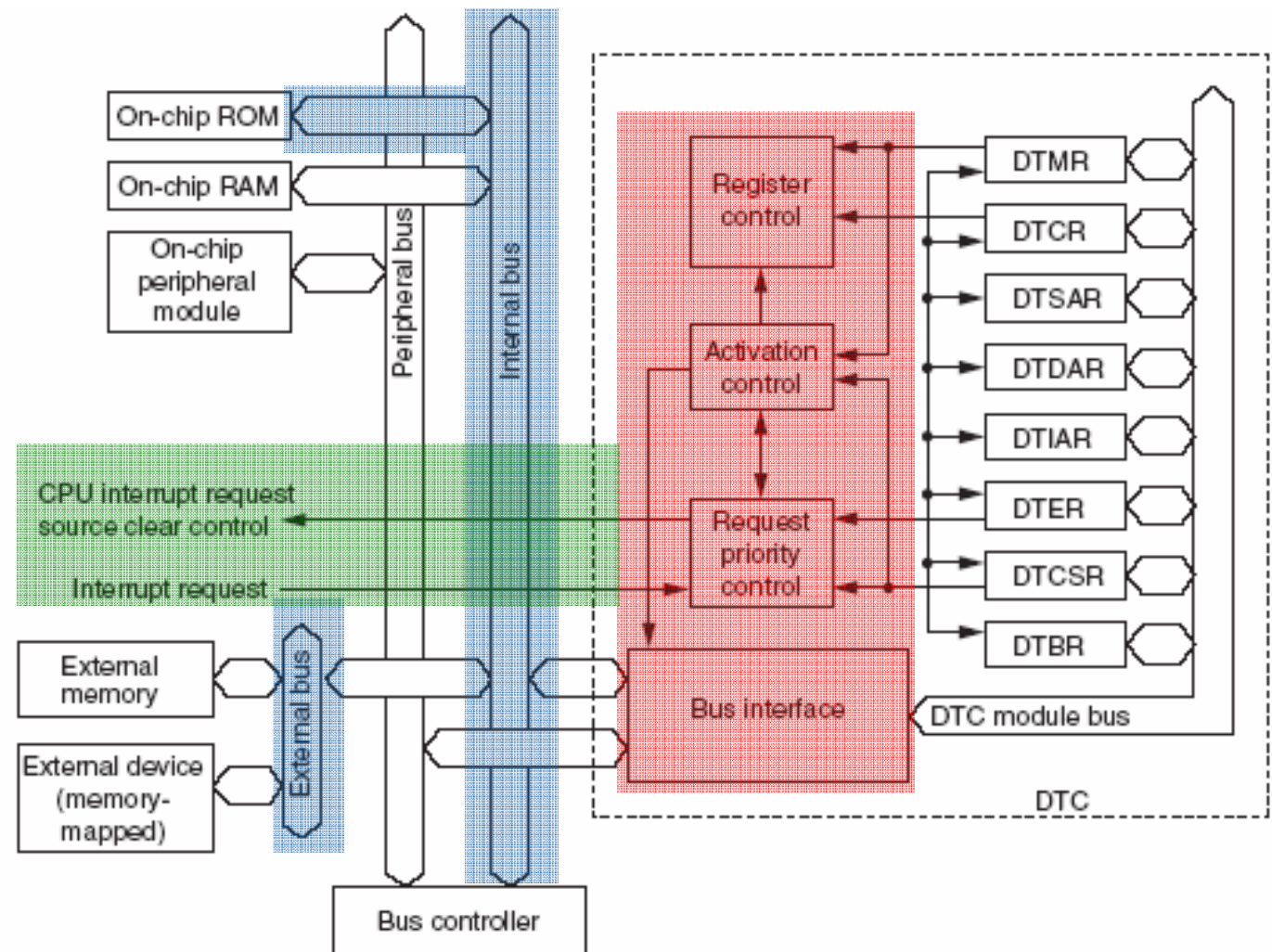


# Renesas SH-2

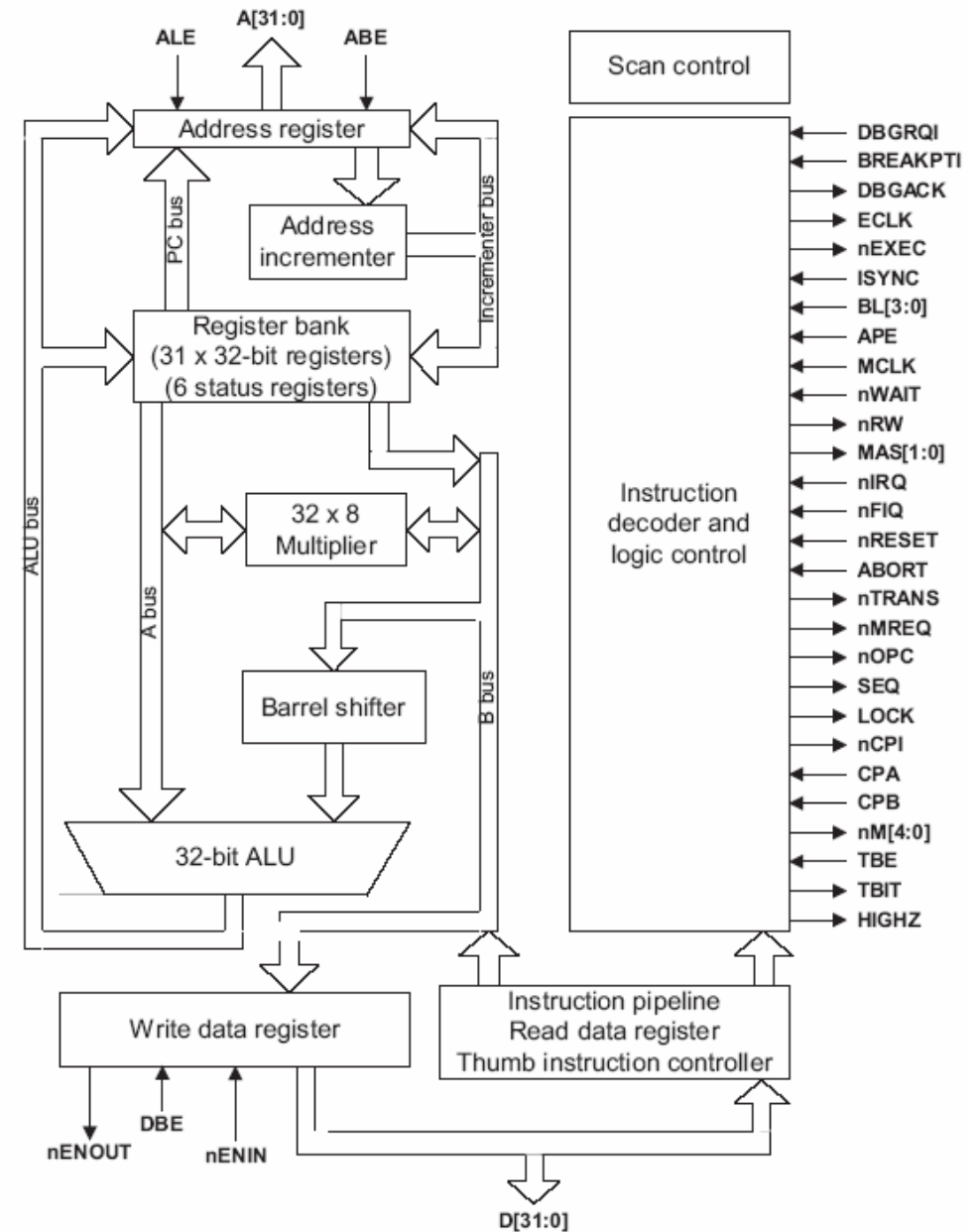
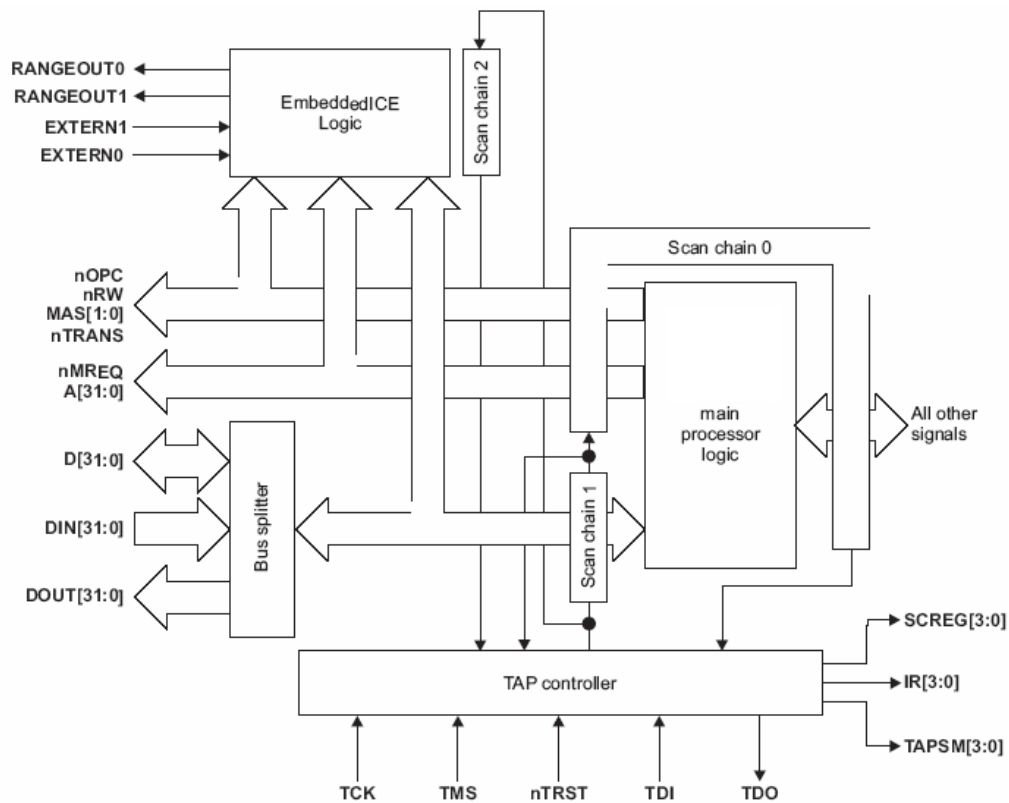




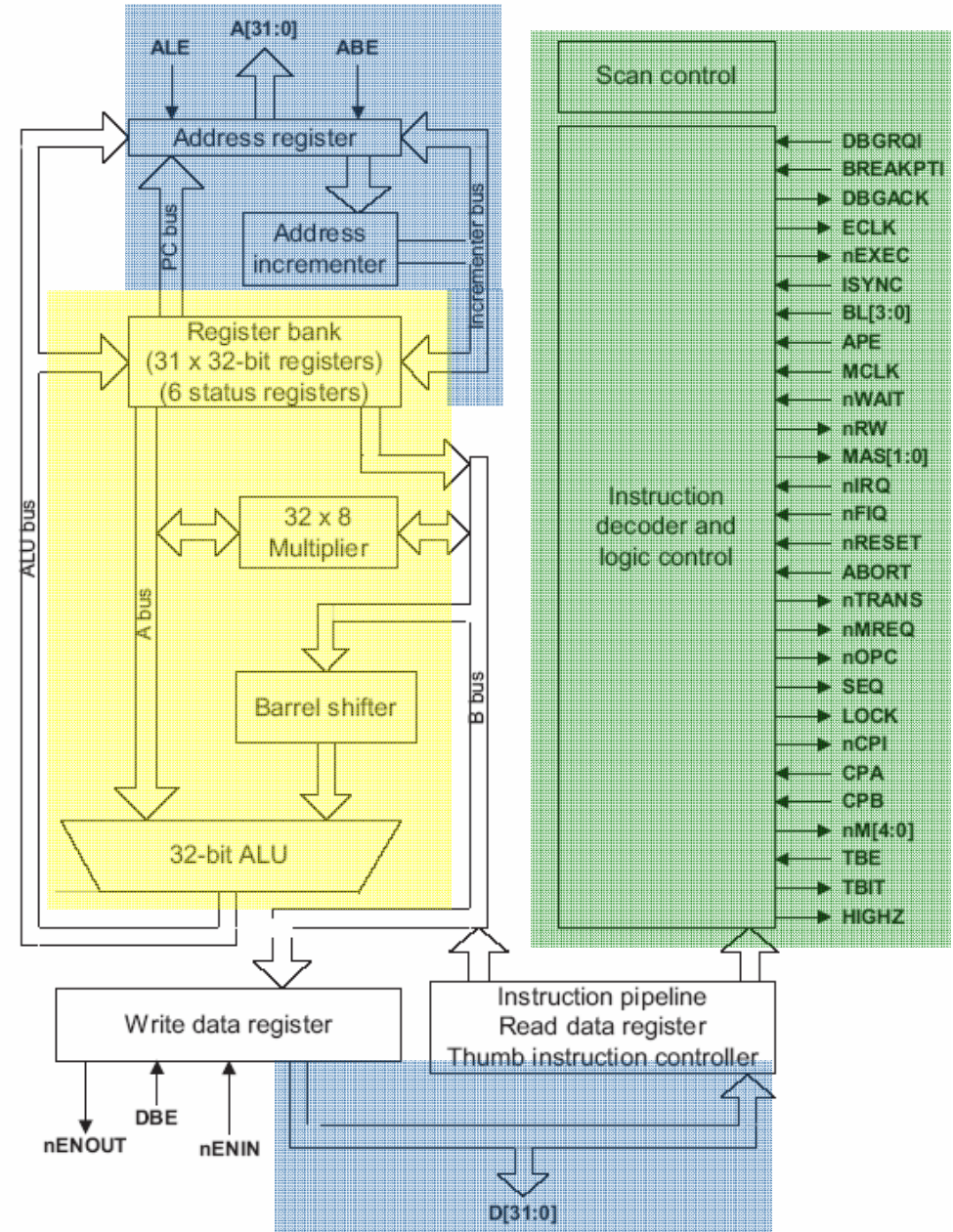
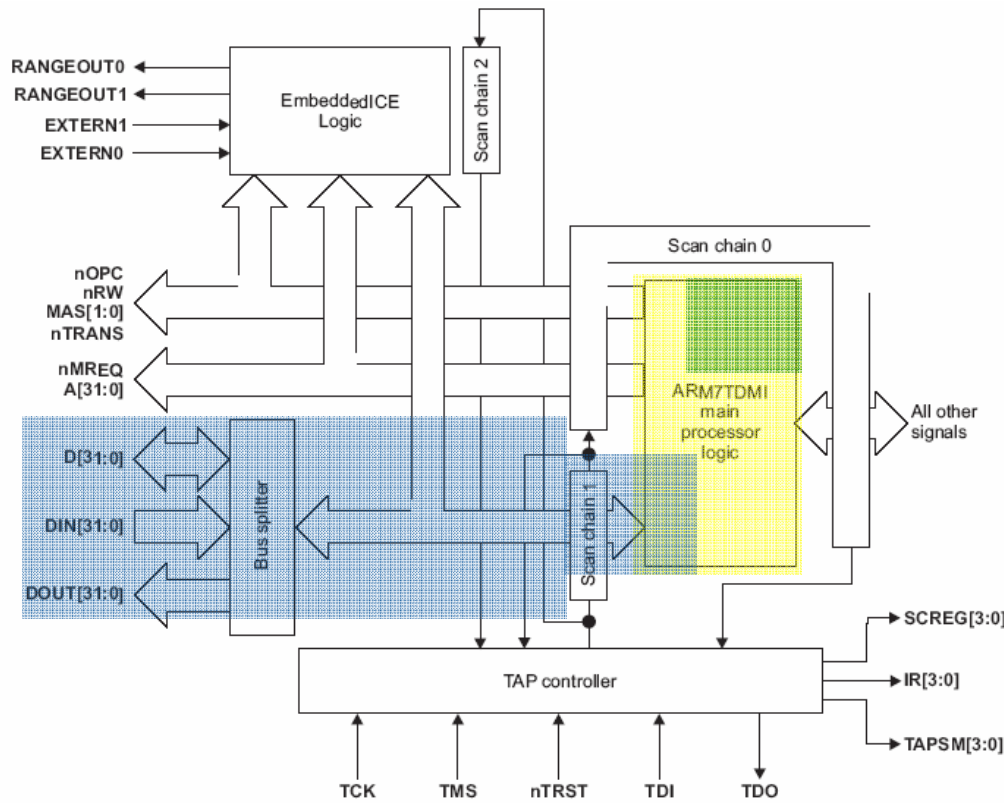
# Renesas SH-2



# Architektura nr. 5



# ARM ARM7TDMI

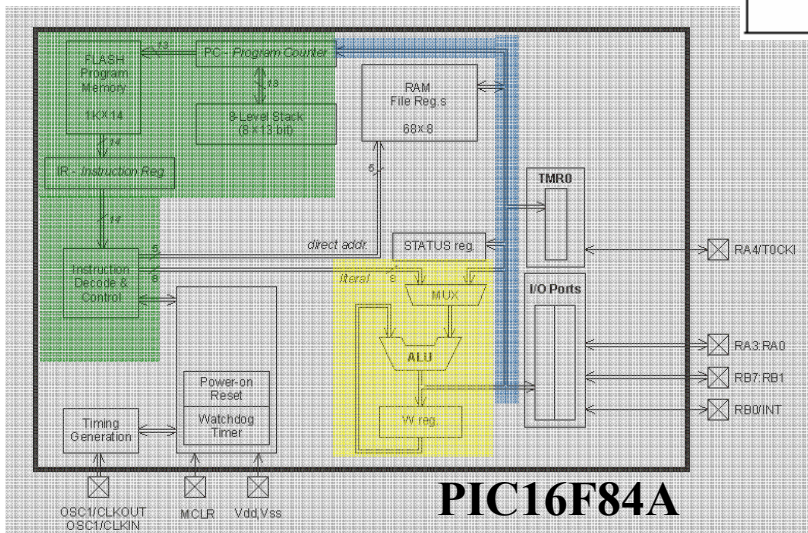
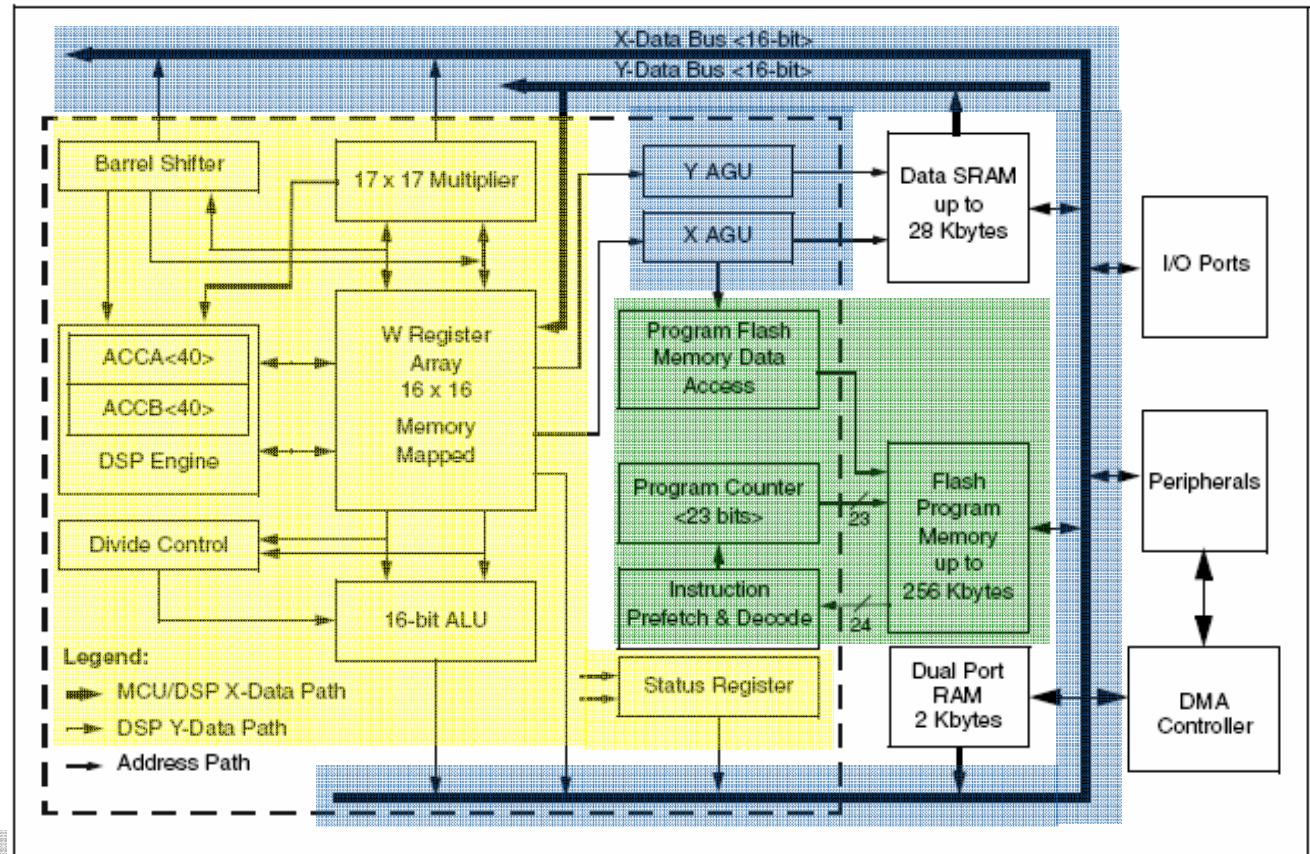


# ARM

## ARM7TDMI

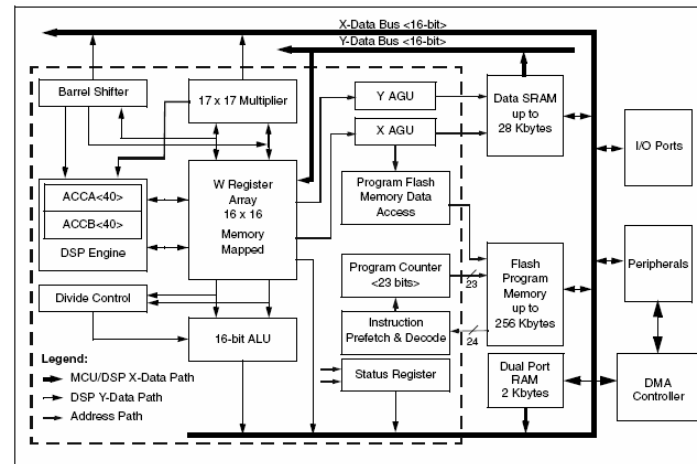
Operation	Assembly syntax	
Move	Move	MOV{cond}{S} Rd, <oprnd2>
	Move NOT	MVN{cond}{S} Rd, <oprnd2>
	Move SPSR to register	MRS{cond} Rd, SPSR
	Move CPSR to register	MRS{cond} Rd, CPSR
	Move register to SPSR	MSR{cond} SPSR{field}, Rm
	Move register to CPSR	MSR{cond} CPSR{field}, Rm
	Move immediate to SPSR flags	MSR{cond} SPSR_f, #32bit_Imm
	Move immediate to CPSR flags	MSR{cond} CPSR_f, #32bit_Imm
Arithmetic	Add	ADD{cond}{S} Rd, Rn, <oprnd2>
	Add with carry	ADC{cond}{S} Rd, Rn, <oprnd2>
	Subtract	SUB{cond}{S} Rd, Rn, <oprnd2>
	Subtract with carry	SBC{cond}{S} Rd, Rn, <oprnd2>
	Subtract reverse subtract	RSB{cond}{S} Rd, Rn, <oprnd2>
	Subtract reverse subtract with carry	RSC{cond}{S} Rd, Rn, <oprnd2>
	Multiply	MUL{cond}{S} Rd, Rm, Rs
	Multiply accumulate	MLA{cond}{S} Rd, Rm, Rs, Rn
Multiply unsigned long	UMULL{cond}{S} RdLo, RdHi, Rm, Rs	

# Architektura nr. 6

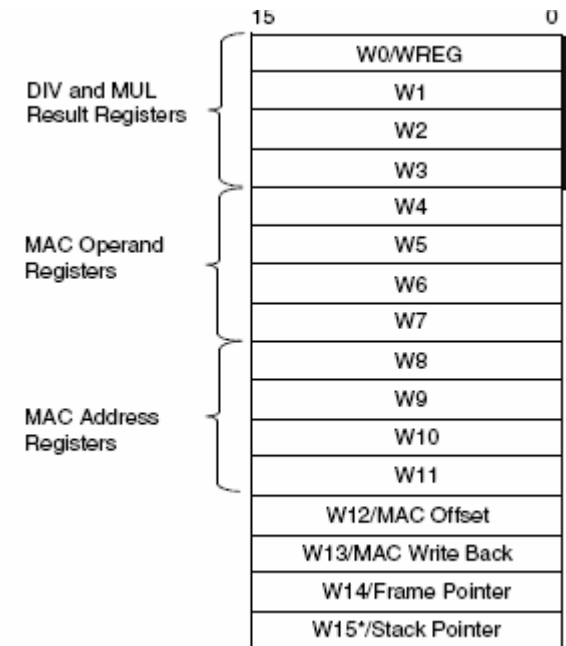




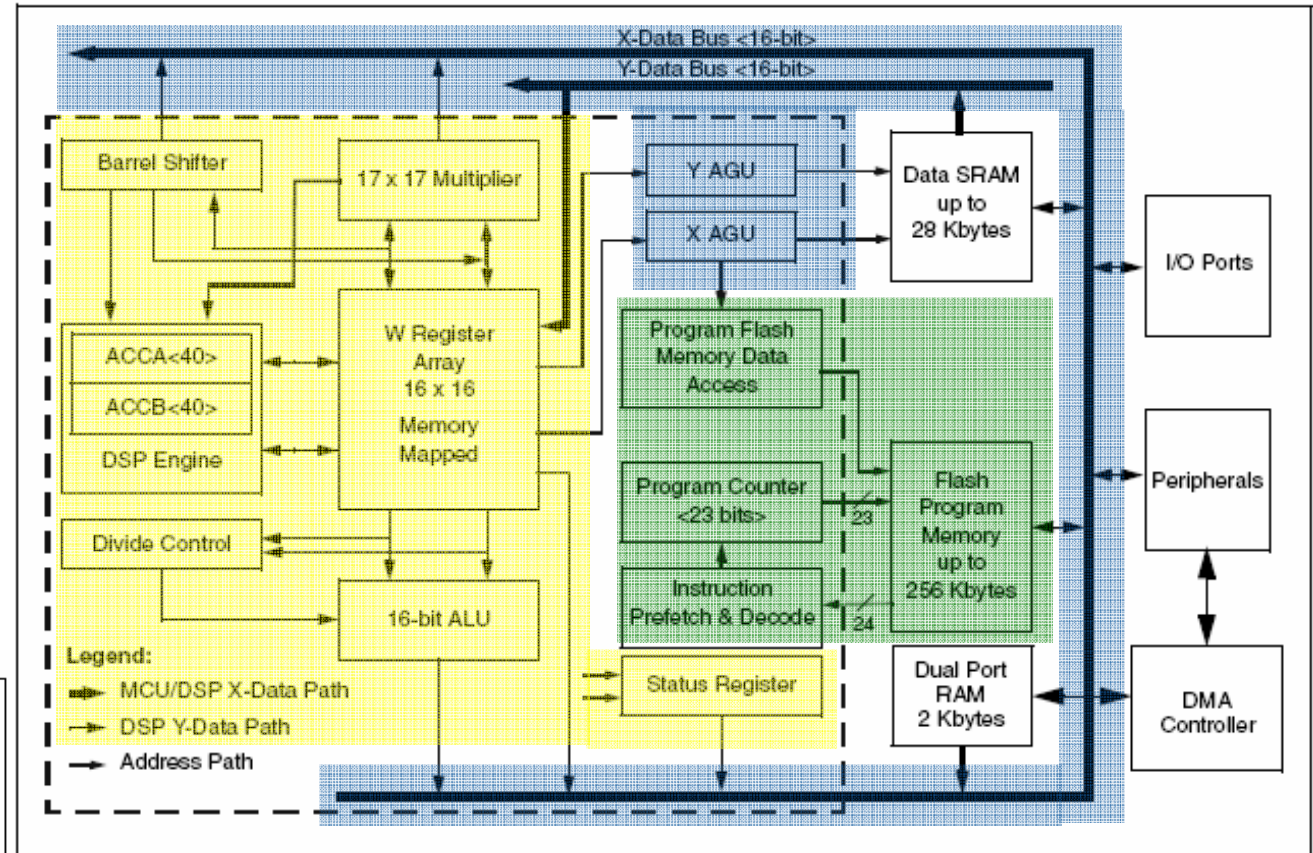
# Microchip dsPIC33F



Assembly	Syntax	Description	Words	Cycles
ADD	Acc	Add accumulators	1	1
ADD	Ws, #Slit4, Acc	16-bit signed add to Acc	1	1
CLR	Acc, Wx, Wxd, Wy, Wyd, AWB	Clear Acc	1	1
ED	Wm*Wm, Acc, Wx, Wy, Wxd	Euclidean distance (no accumulate)	1	1
EDAC	Wm*Wm, Acc, Wx, Wy, Wxd	Euclidean distance	1	1
LAC	Ws, #Slit4, Acc	Load Acc	1	1
MAC	Wm*Wn, Acc, Wx, Wxd, Wy, Wyd, AWB	Multiply and accumulate	1	1
MAC	Wm*Wm, Acc, Wx, Wxd, Wy, Wyd	Square and accumulate	1	1
MOVSAC	Acc, Wx, Wxd, Wy, Wyd, AWB	Move Wx to Wxd and Wy to Wyd	1	1
MPY	Wm*Wn, Acc, Wx, Wxd, Wy, Wyd	Multiply Wn by Wm to Acc	1	1
MPY	Wm*Wm, Acc, Wx, Wxd, Wy, Wyd	Square to Acc	1	1
MPY.N	Wm*Wn, Acc, Wx, Wxd, Wy, Wyd	-(Multiply Wn by Wm) to Acc	1	1
MSC	Wm*Wn, Acc, Wx, Wxd, Wy, Wyd, AWB	Multiply and subtract from Acc	1	1
NEG	Acc	Negate Acc	1	1
SAC	Acc, #Slit4, wd	Store Acc	1	1
SAC.R	Acc, #Slit4, wd	Store rounded Acc	1	1
SFTAC	Acc, #Slit6	Arithmetic shift Acc by Slit6	1	1
SFTAC	Acc, Wn	Arithmetic shift Acc by (Wn)	1	1
SUB	Acc	Subtract accumulators	1	1

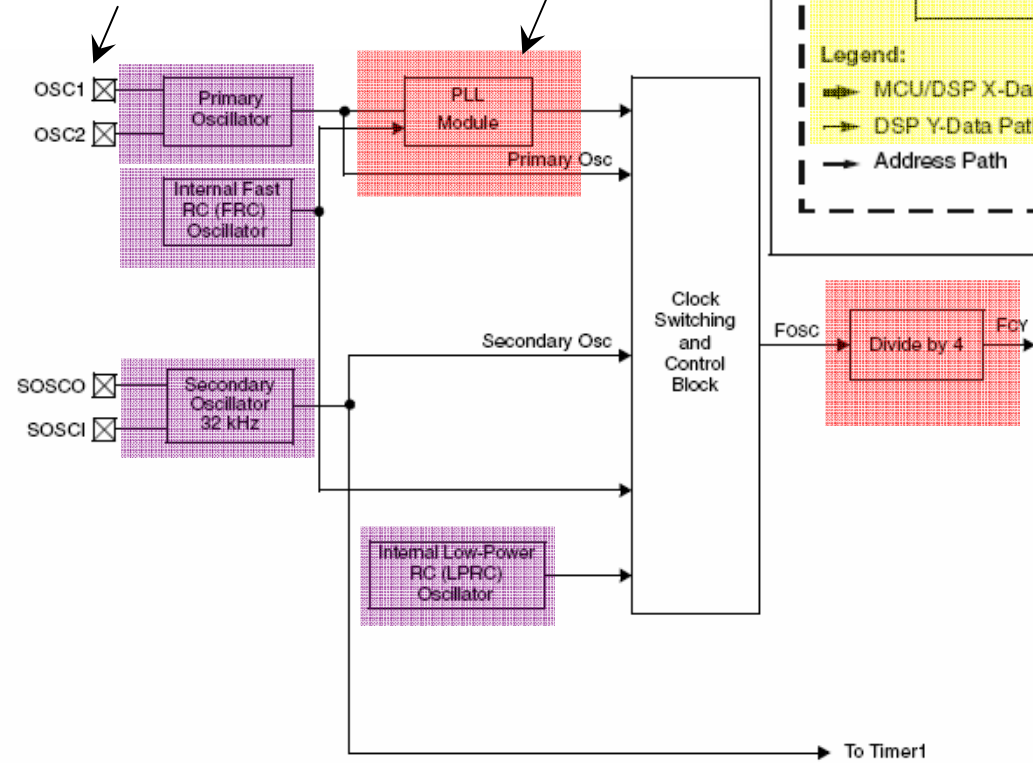


# Microchip dsPIC33F



1.6MHz do 16MHz

???



do 40 MIPS

**KONIEC**