Popis instrukční sady procesoru ADOP

ČVUT FEL, 2008 K. Koubek, P. Bulena

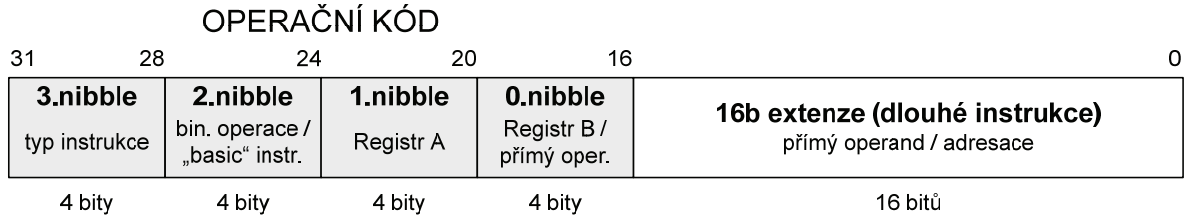
Obsah

Popis instrukční sady

Instrukční sada procesoru ADOP standardně podporuje dvě délky instrukčního kódu – instrukce 16 bitové a 32 bitové. Operační kód je 16 bitový, stojí na počátku instrukce a dělí se na čtyři části, tzv. půlslabiky neboli *nibbles*. 16 bitů následujících za operačním kódem slouží jako přímý operand u dlouhých instrukcí.

Instrukční sada je typu GPR ISA, Registr-Paměť (2,1), tedy ALU instrukce se účastní 2 operandy, přičemž jeden z nich může být z paměti. K použití je až 16 univerzálních registrů, z nichž dva jsou speciální (Zero Register a Stack Pointer). Zero registr obsahuje konstantní nulu, jeho hodnota nemůže být programově přepsána. SP je ukazatel na vrchol zásobníku v paměti, přístup k němu je stejný jako k univerzálnímu registru.

ALU instrukce podporují široké spektrum adresních módů – registrový, přímý operand, registrový nepřímý, bázovaný s konstantou. ISA procesoru ADOP je tak velmi ortogonální. Instrukční sada ADOPu podporuje též 3-operandové instrukce. Umožňuje tedy provádět operace bez přepsání jednoho zdrojového operandu.



Obrázek 1 : Formát ADOP instrukcí

Univerzální registry

R15	R14	R13	R12	R11	R10	R9	R8	R7	R6	R5	R4	R3	R2	R1	R0
SP	registry k všeobecnému použití														ZR

ZR (Zero Register) – registr s konstantní nulou
SP (Stack Pointer) – ukazatel na vrchol zásobníku

Registr příznaků FR

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
-	-	-	-	-	-	-	IF	-	-	-	OF	SF	ZF	AF	CF

Standardní význam příznaků

Pokud nebude dále uvedeno jinak, je význak příznaků v příznakovém registru FR následující:

- **Carry Flag (CF)** – indikuje, že výsledek ALU operace přesahuje velikost řádové mřížky. Odpovídá nejvyššímu bitu rozšířené 17-bitové datové cesty uvnitř ALU procesoru ADOP.
- **Auxiliary Flag (AF)** – používá se k aritmetickým operacím nad slovy různých délek. Je určen nejvyšším bitem operandu B. Jeho extenzí na 16 bitů získáme tzv. XAF (Extended AF) – ten zastupuje kratší slovo při dalších krocích výpočtu.
- **Zero Flag (ZF)** – indikuje, že výsledek ALU operace je nula. Je nastaven na jedničku, pokud jsou všechny bity výsledku nulové.
- **Sign Flag (SF)** – udává, zda je výsledek ALU operace záporný. V doplňkovém kódu je určen nejvyšším bitem výsledku.
- **Overflow Flag (OF)** – má podobnou funkci jako carry flag, ovšem pro čísla se znaménkem. OF je nastaven, pokud znaménko výsledku sčítání nebo odčítání neodpovídá očekávání (např. součet dvou kladných čísel dá záporný výsledek).
- **Interrupt Flag (IF)** – určuje, zda jsou povolena vnější přerušení. Pokud je IF roven jedné, jsou přerušení povolena, v opačném případě jsou přerušení zakázána.

Přehled instrukcí

Nyní následuje stručný přehled všech na ADOPu podporovaných instrukcí.

ADD – Add

Opcode	Instrukce	Popis
1 1 *rA imm4*	ADD *rA*, *imm4*	Přičtení *imm4* k *rA*
2 1 *rA rB*	ADD *rA*, *rB*	Přičtení *rB* k *rA*
3 1 *rA* x	ADD *rA*, XAF	Přičtení XAF k *rA*
4 1 *rA rB*	ADDB *rA*, [*rB*]	Přičtení 8-bit operandu bez znaménka adresovaného v paměti *rB* k *rA*
5 1 *rA rB*	ADDS *rA*, [*rB*]	Přičtení 8-bit operandu se znaménkem adresovaného v paměti *rB* k *rA*
6 1 *rA rB*	ADD *rA*, [*rB*]	Přičtení 16-bit operandu adresovaného v paměti *rB* k *rA*
8 1 *rA* x *imm16*	ADD *rA*, *imm16*	Přičtení *imm16* k *rA*
B 1 *rA rB imm16*	ADD *rA*, *rB*, *imm16*	Přičtení *rB* k *imm16*, výsledek uložen do *rA*
C 1 *rA rB offs16*	ADDB *rA*, [*rB*+*offs16*]	Přičtení 8-bit operandu bez znaménka adresovaného v paměti *rB*+*offs16* k *rA*
D 1 *rA rB offs16*	ADDS *rA*, [*rB*+*offs16*]	Přičtení 8-bit operandu se znaménkem adresovaného v paměti *rB*+*offs16* k *rA*
E 1 *rA rB offs16*	ADD *rA*, [*rB*+*offs16*]	Přičtení 16-bit operandu adresovaného v paměti *rB*+*offs16* k *rA*

Popis

Instrukce provádí znaménkové sčítání. Výsledek je platný pro znaménkové i neznaménkové operandy, příznaky OF a CF indikují přetečení/přenos znaménkového/neznaménkového výsledku. Příznak SF indikuje znaménko znaménkového výsledku.

Operace

DEST <- DEST + SRC; nebo

DEST <- SRC1 + SRC2; pro trojoperandový tvar

Příznaky

OF, SF, ZF, AF, CF jsou nastaveny podle pravidel.

ADC – Add with Carry

Opcode	**Instrukce**	**Popis**
1 2 *rA imm4*	ADC *rA*, *imm4*	Přičtení s přenosem *imm4* k *rA*
2 2 *rA rB*	ADC *rA*, *rB*	Přičtení s přenosem *rB* k *rA*
3 2 *rA* x	ADC *rA*, XAF	Přičtení s přenosem XAF k *rA*
4 2 *rA rB*	ADCB *rA*, [*rB*]	Přičtení s přenosem 8-bit operandu bez znaménka adresovaného v paměti *rB* k *rA*
5 2 *rA rB*	ADCS *rA*, [*rB*]	Přičtení s přenosem 8-bit operandu se znaménkem adresovaného v paměti *rB* k *rA*
6 2 *rA rB*	ADC *rA*, [*rB*]	Přičtení s přenosem 16-bit operandu adresovaného v paměti *rB* k *rA*
8 2 *rA* x *imm16*	ADC *rA*, *imm16*	Přičtení s přenosem *imm16* k *rA*
B 2 *rA rB imm16*	ADC *rA*, *rB*, *imm16*	Přičtení s přenosem *rB* k *imm16*, výsledek uložen do *rA*
C 2 *rA rB offs16*	ADCB *rA*, [*rB*+*offs16*]	Přičtení s přenosem 8-bit operandu bez znaménka adresovaného v paměti *rB*+*offs16* k *rA*
D 2 *rA rB offs16*	ADCS *rA*, [*rB*+*offs16*]	Přičtení s přenosem 8-bit operandu se znaménkem adresovaného v paměti *rB*+*offs16* k *rA*
E 2 *rA rB offs16*	ADC *rA*, [*rB*+*offs16*]	Přičtení s přenosem 16-bit operandu adresovaného v paměti *rB*+*offs16* k *rA*

Popis

Instrukce provádí znaménkové sčítání dvou operandů a CF. Výsledek je platný pro znaménkové i neznaménkové operandy, příznaky OF a CF indikují přetečení/přenos znaménkového/neznaménkového výsledku. Příznak SF indikuje znaménko znaménkového výsledku.

Operace

DEST <- DEST + SRC + CF; nebo

DEST <- SRC1 + SRC2 + CF; pro trojoperandový tvar

Příznaky

OF, SF, ZF, AF, CF jsou nastaveny podle pravidel.

AND – Logical AND

Opcode	**Instrukce**	**Popis**
1 5 *rA imm4*	AND *rA*, *imm4*	*imm4* AND *rA*
2 5 *rA rB*	AND *rA*, *rB*	*rB* AND *rA*
3 5 *rA* x	AND *rA*, XAF	XAF AND *rA*
4 5 *rA rB*	ANDB *rA*, [*rB*]	8-bit operand bez znaménka adresovaný v paměti *rB* AND *rA*
5 5 *rA rB*	ANDS *rA*, [*rB*]	8-bit operand se znaménkem adresovaný v paměti *rB* AND *rA*
6 5 *rA rB*	AND *rA*, [*rB*]	16-bit operand adresovaný v paměti *rB* AND *rA*
8 5 *rA* x *imm16*	AND *rA*, *imm16*	*Imm16* AND *rA*
B 5 *rA rB imm16*	AND *rA*, *rB*, *imm16*	*rB* AND *imm16*, výsledek uložen do *rA*
C 5 *rA rB offs16*	ANDB *rA*, [*rB*+*offs16*]	8-bit operand bez znaménka adresovaný v paměti *rB*+*offs16* AND *rA*
D 5 *rA rB offs16*	ANDS *rA*, [*rB*+*offs16*]	8-bit operand se znaménkem adresovaný v paměti *rB*+*offs16* AND *rA*
E 5 *rA rB offs16*	AND *rA*, [*rB*+*offs16*]	16-bit operand adresovaný v paměti *rB*+*offs16* AND *rA*

Popis

Instrukce provádí operaci AND mezi jednotlivými bity operandů.

Operace

DEST <- DEST AND SRC; nebo

DEST <- SRC1 AND SRC2; pro trojoperandový tvar

Příznaky

OF není definován, CF je vynulován. AF, SF a ZF jsou nastaveny podle pravidel.

ASR – Arithmetic Shift Right

Opcode	**Instrukce**	**Popis**
1 B *rA imm4*	ASR *rA*, *imm4*	Aritmetický posun *rA* vpravo o *imm4*
2 B *rA rB*	ASR *rA*, *rB*	Aritmetický posun *rA* vpravo o *rB*
3 B *rA* x	ASR *rA*, XAF	Aritmetický posun *rA* vpravo o XAF
4 B *rA rB*	ASRB *rA*, [*rB*]	Aritmetický posun *rA* vpravo o 8-bit operand bez znaménka adresovaný v paměti *rB*
6 B *rA rB*	ASR *rA*, [*rB*]	Aritmetický posun *rA* vpravo o 16-bit operand adresovaný v paměti *rB*
8 B *rA* x *imm16*	ASR *rA*, *imm16*	Aritmetický posun *rA* vpravo o *imm16*
B B *rA rB imm16*	ASR *rA*, *rB*, *imm16*	Aritmetický posun *rB* vpravo o *imm16*, výsledek uložen do *rA*
C B *rA rB offs16*	ASRB *rA*, [*rB+offs16*]	Aritmetický posun *rA* vpravo o 8-bit operand bez znaménka adresovaný v paměti *rB+offs16*
E B *rA rB offs16*	ASR *rA*, [*rB+offs16*]	Aritmetický posun *rA* vpravo o 16-bit operand adresovaný v paměti *rB+offs16*

Popis

Instrukce provádí aritmetický posun bitů prvního operandu vpravo, zleva je kopírován nejvyšší bit tohoto operandu.

Operace

DEST <- DEST >>> SRC; nebo

DEST <- SRC1 >>> SRC2; pro trojoperandový tvar

Příznaky

Příznak CF obsahuje poslední vysunutý bit. OF není definován. SF, ZF a AF jsou nastaveny podle pravidel.

CALL – Call Procedure

Opcode	Instrukce	Popis
A 0 x x *offs16*	CALL *offs16*	Volání podprogramu na relativní adrese *offs16*

Popis

Instrukce uloží adresu následující instrukce (PC+4) na zásobník a provede volání podprogramu na relativní adrese *offs16*.

Operace

r15 <- r15 – 2;

[r15] <- PC + 4;

PC <- PC + 2 + offs16;

Příznaky

Nezměněny.

CALLA – Call Procedure Absolute

Opcode	**Instrukce**	**Popis**
A 1 x x *offs16*	CALLA *offs16*	Volání podprogramu na absolutní adrese *offs16*

Popis

Instrukce uloží adresu následující instrukce (PC+4) na zásobník a provede volání podprogramu na absolutní adrese *offs16*.

Instrukce umožňuje volání systémových služeb přes tabulku, kdy číslo služby násobené čtyřmi určuje adresu offs16.

Operace

r15 <- r15 – 2;

[r15] <- PC + 4;

PC <- offs16;

Příznaky

Nezměněny.

CALLI – Call Procedure Indirect

Opcode	**Instrukce**	**Popis**
0 1 *rA* 1	CALLI *rA*	Volání podprogramu, jehož adresa je v *rA*

Popis

Instrukce uloží adresu následující instrukce (PC+2) na zásobník a provede volání podprogramu na absolutní adrese v *rA*.

Operace

r15 <- r15 – 2;

[r15] <- PC + 2;

PC <- rA;

Příznaky

Nezměněny.

CCB – Clear Cache Block

Opcode	**Instrukce**	**Popis**
7 8 x *rB*	CCB [*rB*]	Zneplatnění bloku na adrese *rB*
F 8 x *rB*	CCB [*rB*+*offs16*]	Zneplatnění bloku na adrese *rB*+*offs16*

Popis

Instrukce zneplatní blok instrukční a datové cache na určené adrese. Valid bit příslušného tagu je nulován.

Operace

VB <- 0;

Příznaky

Nezměněny.

CLC – Clear Carry Flag

Opcode	Instrukce	Popis
0 5 x 0	CLC	Nulování příznaku přenosu

Popis

Instrukce nuluje příznak CF.

Operace

CF <- 0;

Příznaky

CF nulováno, ostatní nezměněny.

CLI – Clear Interrupt Flag

Opcode	Instrukce	Popis
0 5 x 2	CLI	Nulování příznaku povolení přerušení

Popis

Instrukce nuluje příznak IF. Přerušení jsou zakázána.

Operace

IF <- 0;

Příznaky

IF nulováno, ostatní nezměněny.

CMP – Compare Two Operands

Opcode	**Instrukce**	**Popis**
1 8 *rA imm4*	CMP *rA*, *imm4*	Porovnání *imm4* s *rA*
2 8 *rA rB*	CMP *rA*, *rB*	Porovnání *rB* s *rA*
3 8 *rA* x	CMP *rA*, XAF	Porovnání XAF s *rA*
4 8 *rA rB*	CMBS *rA*, [*rB*]	Porovnání 8-bit operandu bez znaménka adresovaného v paměti *rB* s *rA*
5 8 *rA rB*	CMPS *rA*, [*rB*]	Porovnání 8-bit operandu se znaménkem adresovaného v paměti *rB* s *rA*
6 8 *rA rB*	CMP *rA*, [*rB*]	Porovnání 16-bit operandu adresovaného v paměti *rB* s *rA*
8 8 *rA* x *imm16*	CMP *rA*, *imm16*	Porovnání *imm16* s *rA*
B 8 *rA rB imm16*	CMP *rA*, *rB*, *imm16*	Porovnání *imm16* s *rB*; *rA* není použit
C 8 *rA rB offs16*	CMPB *rA*, [*rB*+*offs16*]	Porovnání 8-bit operandu bez znaménka adresovaného v paměti *rB*+*offs16* s *rA*
D 8 *rA rB offs16*	CMPS *rA*, [*rB*+*offs16*]	Porovnání 8-bit operandu se znaménkem adresovaného v paměti *rB*+*offs16* s *rA*
E 8 *rA rB offs16*	CMP *rA*, [*rB*+*offs16*]	Porovnání 16-bit operandu adresovaného v paměti *rB*+*offs16* s *rA*

Popis

Porovnává dané operandy. Porovnání je provedeno odečtením druhého operandu od prvního. Následně jsou podle výsledku nastaveny příznaky.

Operace

temp <- DEST - SRC; nebo
temp <- SRC1 – SRC2; pro trojoperandový tvar

NastaveníPříznaků; (Odpovídá instrukci SUB)

Příznaky

OF, SF, ZF, AF, CF jsou nastaveny podle pravidel.

DEC – Decrement by 1

Opcode	**Instrukce**	**Popis**
0 3 *rA 3*	DEC *rA*	Dekrementace *rA o 1*

Popis

Instrukce odečítá 1 od cílového operandu při zachování stavu CF.

Operace

DEST <- DEST - 1;

Příznaky

OF, SF, ZF, AF, CF jsou nastaveny podle pravidel.

INC – Increment by 1

Opcode	**Instrukce**	**Popis**
0 3 *rA 2*	INC *rA*	Inkrementace *rA o 1*

Popis

Instrukce přičítá 1 k cílovému operandu při zachování stavu CF.

Operace

DEST <- DEST + 1;

Příznaky

OF, SF, ZF, AF, CF jsou nastaveny podle pravidel.

IRC – Interrupt Routine Call

Opcode	Instrukce	Popis
A 8 0 x *offs16*		Volání obsluhy HW přerušení na absolutní adrese *offs16*

Popis

Instrukce uloží adresu aktuální instrukce (PC) na zásobník, zakáže vnější přerušení a provede volání podprogramu na absolutní adrese *offs16*. Instrukce je vynucena při vyvolání vnějšího přerušení.

Jedná se o vnitřní instrukci procesoru, programátorovi není dostupná.

Operace

r15 <- r15 – 2;

[r15] <- PC;

IF <- 0;

PC <- offs16;

Příznaky

IF nulován, ostatní nezměněny.

JCC – Jump if Condition is Met

Opcode	**Instrukce**	**Popis**
9 0 0 x *offs16*	JZ *offs16*	Skok když ZF=1
9 1 0 x *offs16*	JNZ *offs16*	Skok když ZF=0
9 2 0 x *offs16*	JC *offs16*	Skok když CF=1
9 3 0 x *offs16*	JNC *offs16*	Skok když CF=0
9 4 0 x *offs16*	JS *offs16*	Skok když SF=1
9 5 0 x *offs16*	JNS *offs16*	Skok když SF=0
9 6 0 x *offs16*	JO *offs16*	Skok když OF=1
9 7 0 x *offs16*	JNO *offs16*	Skok když OF=0
9 8 0 x *offs16*	JGE *offs16*	Skok když větší nebo rovno (SF=OF)
9 9 0 x *offs16*	JLT *offs16*	Skok když menší (SF != OF)
9 A 0 x *offs16*	JGT *offs16*	Skok když větší (ZF=0 & SF=OF)
9 B 0 x *offs16*	JLE *offs16*	Skok když menší nebo rovno (ZF=1 | SF!=OF)
9 C 0 x *offs16*	JAT *offs16*	Skok když nad (CF=0 & ZF=0)
9 D 0 x *offs16*	JBE *offs16*	Skok když pod nebo rovno (CF=1 | ZF=1)
9 E 0 x *offs16*	JMP *offs16*	Nepodmíněný skok, viz instrukce JMP
9 3 0 x *offs16*	JAE *offs16*	Skok když nad nebo rovno (CF=0)
9 2 0 x *offs16*	JBT *offs16*	Skok když pod (CF=1)
9 0 0 x *offs16*	JEQ *offs16*	Skok když rovno (ZF=1)
9 1 0 x *offs16*	JNE *offs16*	Skok když nerovno (ZF=0)

Popis

Provede relativní skok dle vyhodnocení podmínky.

Operace

IF COND THEN PC <- PC + 2 + offs16;

Příznaky

Nezměněny.

JMP – Jump

Opcode	Instrukce	Popis
9 E 0 x *offs16*	JMP *offs16*	Relativní skok s 16-bit offsetem

Popis

Provede relativní skok.

Operace

PC <- PC + 2 + offs16;

Příznaky

Nezměněny.

JMPA – Jump Absolute

Opcode	Instrukce	Popis
9 F 0 x *offs16*	JMPA *offs16*	Absolutní skok na adresu *offs16*

Popis

Provede absolutní skok.

Operace

PC <- offs16;

Příznaky

Nezměněny.

JMPI – Jump Indirect

Opcode	**Instrukce**	**Popis**
0 1 *rA* 0	JMPI *rA*	Nepřímý skok přes registr *rA*

Popis

Provede nepřímý skok na absolutní adresu v *rA*.

Operace

PC <- rA;

Příznaky

Nezměněny.

MHR – Move Hidden Register

Opcode	**Instrukce**	**Popis**
0 A *rA hrB*	MHR *rA*, *hrB*	Zkopíruje obsah skrytého registru *hrB* do registru *rA*

Popis

Instrukce kopíruje obsah skrytého registru hrB do registru rA. Využití má ve spojení s instrukcí MUL pro uložení horních šestnácti bitů výsledku.

Operace

DEST <- SRC;

Příznaky

Nezměněny.

MOV – Move

Opcode	Instrukce	Popis
1 0 *rA imm4*	MOV *rA*, *imm4*	Nahraje *imm4* do *rA*
2 0 *rA rB*	MOV *rA*, *rB*	Nahraje *rB* do *rA*
3 0 *rA* x	MOV *rA*, XAF	Nahraje XAF do *rA*
4 0 *rA rB*	MOVB *rA*, [*rB*]	Nahraje 8-bit operand bez znaménka adresovaný v paměti *rB* do *rA*
5 0 *rA rB*	MOVS *rA*, [*rB*]	Nahraje 8-bit operand se znaménkem adresovaný v paměti *rB* do *rA*
6 0 *rA rB*	MOV *rA*, [*rB*]	Nahraje 16-bit operand adresovaný v paměti *rB* do *rA*
8 0 *rA* x *imm16*	MOV *rA*, *imm16*	Nahraje *imm16* do *rA*
C 0 *rA rB offs16*	MOVB *rA*, [*rB*+*offs16*]	Nahraje 8-bit operand bez znaménka adresovaný v paměti *rB*+*offs16* do *rA*
D 0 *rA rB offs16*	MOVS *rA*, [*rB*+*offs16*]	Nahraje 8-bit operand se znaménkem adresovaný v paměti *rB*+*offs16* do *rA*
E 0 *rA rB offs16*	MOV *rA*, [*rB*+*offs16*]	Nahraje 16-bit operand adresovaný v paměti *rB*+*offs16* do *rA*

Popis

Instrukce zkopíruje hodnotu zdrojového operandu do registru rA.

Operace

DEST <- SRC;

Příznaky

Nezměněny.

MULS – Signed Multiply

Opcode	Instrukce	Popis
0 9 *rA* rB	MULS *rA*, *rB*	Znaménkové vynásobení *rA* hodnotou *rB*

Popis

Instrukce znaménkově vynásobí hodnoty v registrech rA a rB. Spodních 16 bitů výsledku uloží do registru rA, horních 16 bitů výsledku do skrytého registru HR0.

Operace

DEST <- (DEST * SRC) AND 0xFFFF;

HR0 <- (DEST*SRC) >> 16;

Příznaky

CF je nulován, pokud je všech horních 16 bitů výsledku rovno buď 0 nebo 1, jinak je nastaven na 1. ZF a SF jsou nastaveny podle spodních 16 bitů výsledku. OF a AF nejsou definovány.

MULU – Unsigned Multiply

Opcode	Instrukce	Popis
0 8 *rA* rB	MULU *rA*, *rB*	Neznaménkové vynásobení *rA* hodnotou *rB*

Popis

Instrukce neznaménkově vynásobí hodnoty v registrech rA a rB. Spodních 16 bitů výsledku uloží do registru rA, horních 16 bitů výsledku do skrytého registru HR0.

Operace

DEST <- (DEST * SRC) AND 0xFFFF;

HR0 <- (DEST*SRC) >> 16;

Příznaky

CF je nulován, pokud je všech horních 16 bitů výsledku rovno 0, jinak je nastaven na 1. ZF a SF jsou nastaveny podle spodních 16 bitů výsledku. OF a AF nejsou definovány.

NEG – Two's Complement Negation

Opcode	Instrukce	Popis
0 3 *rA* 1	NEG *rA*	Dvojkový doplněk rA

Popis

Nahradí hodnotu operandu jeho dvojkovým doplňkem (tato operace je ekvivalentní odečtení operandu od nuly).

Operace

DEST <- 0 - DEST;

Příznaky

Příznak CF je nastaven na 0, jestliže je operand nulový; jinak je CF nastaven na 1. Příznaky OF, ZF, SF jsou nastaveny dle pravidel. Příznak AF není definován.

NOP – No Operation

Opcode	Instrukce	Popis
0 0 x x	NOP	Žádná operace

Popis

Instrukce neovlivňuje stav procesoru kromě PC.

Operace

Žádná

Příznaky

Nezměněny.

NOT – One's Complement Negation

Opcode	Instrukce	Popis
0 3 *rA* 0	NOT *rA*	Negace každého bitu *rA*

Popis

Provádí bitovou negaci (každá 1 je nastavena na 0, každá 0 nastavena na 1) na operandu v rA.

Operace

DEST <- not DEST;

Příznaky

Nezměněny.

OR – Logical Inclusive OR

Opcode	**Instrukce**	**Popis**
1 6 *rA imm4*	OR *rA*, *imm4*	*imm4* OR *rA*
2 6 *rA rB*	OR *rA*, *rB*	*rB* OR *rA*
3 6 *rA* x	OR *rA*, XAF	XAF OR *rA*
4 6 *rA rB*	ORB *rA*, [*rB*]	8-bit operand bez znaménka adresovaný v paměti *rB* OR *rA*
5 6 *rA rB*	ORS *rA*, [*rB*]	8-bit operand se znaménkem adresovaný v paměti *rB* OR *rA*
6 6 *rA rB*	OR *rA*, [*rB*]	16-bit operand adresovaný v paměti *rB* OR *rA*
8 6 *rA* x *imm16*	OR *rA*, *imm16*	*imm16* OR *rA*
B 6 *rA rB imm16*	OR *rA*, *rB*, *imm16*	*imm16* OR *rB*, výsledek uložen do *rA*
C 6 *rA rB offs16*	ORB *rA*, [*rB*+*offs16*]	8-bit operand bez znaménka adresovaný v paměti *rB*+*offs16* OR *rA*
D 6 *rA rB offs16*	ORS *rA*, [*rB*+*offs16*]	8-bit operand se znaménkem adresovaný v paměti *rB*+*offs16* OR *rA*
E 6 *rA rB offs16*	OR *rA*, [*rB*+*offs16*]	16-bit operand adresovaný v paměti *rB*+*offs16* OR *rA*

Popis

Instrukce provádí operaci OR mezi jednotlivými bity operandů.

Operace

DEST <- DEST OR SRC; nebo

DEST <- SRC1 OR SRC2; pro trojoperandový tvar

Příznaky

OF a CF jsou vynulovány, AF, SF a ZF jsou nastaveny podle pravidel.

POP – Pop a Value from the Stack

Opcode	**Instrukce**	**Popis**
0 4 *rA* 2	POP *rA*	Načtení hodnoty ze zásobníku do *rA*

Popis

Načte hodnotu z vrcholu zásobníku do registru rA a k ukazateli zásobníku (r15) přičte 2.

Operace

rA <- [r15];
r15 <- r15 + 2;

Příznaky

Nezměněny.

POPF – Pop Stack into FLAGS Register

Opcode	Instrukce	Popis
0 4 x 3	POPF	Načtení hodnoty ze zásobníku do příznakového registru

Popis

Načte hodnotu z vrcholu zásobníku do FLAGS registru a zvýší hodnotu ukazatele zásobníku (r15) o 2.

Operace

FLAGS <- [r15];
r15 <- r15 + 2;

Příznaky

Přepsány načtenou hodnotou.

PUSH – Push Word Onto the Stack

Opcode	Instrukce	Popis
0 4 *rA* 0	PUSH *rA*	Uložení *rA* na zásobník

Popis

Uloží obsah registru rA na zásobník a zmenší hodnotu ukazatele zásobníku (r15) o 2.

Operace

r15 <- r15 – 2;

[r15] <- rA;

Příznaky

Nezměněny.

PUSHF – Push Flags Register Onto the Stack

Opcode	Instrukce	Popis
0 4 x 1	PUSHF	Uložení registru příznaků na zásobník

Popis

Uloží obsah registru příznaků na zásobník a sníží hodnotu ukazatele zásobníku (r15) o 2.

Operace

r15 <- r15 – 2;

[r15] <- FLAGS;

Příznaky

Nezměněny.

RET – Return from Procedure

Opcode	Instrukce	Popis
0 2 x 0	RET	Návrat z procedury

Popis

Ze zásobníku načte hodnotu a použije ji jako nový obsah PC.

Operace

PC <- [r15];
r15 <- r15 + 2;

Příznaky

Nezměněny.

RETI – Return from Interrupt

Opcode	Instrukce	Popis
0 2 x 1	RETI	Návrat z obsluhy přerušení

Popis

Ze zásobníku načte hodnotu a použije ji jako nový obsah PC. Povolí přerušení.

Operace

PC <- [r15];
r15 <- r15 + 2;
IF <- 1;

Příznaky

IF nastaven na 1. Ostatní nezměněny.

RLC – Rotate Left through Carry

Opcode	**Instrukce**	**Popis**
1 D *rA imm4*	RLC *rA*, *imm4*	Rotace *rA* vlevo přes přenos o *imm4*
2 D *rA rB*	RLC *rA*, *rB*	Rotace *rA* vlevo přes přenos o *rB*
3 D *rA* x	RLC *rA*, XAF	Rotace *rA* vlevo přes přenos o XAF
4 D *rA rB*	RLCB *rA*, [*rB*]	Rotace *rA* vlevo přes přenos o 8-bit operand bez znaménka adresovaný v paměti *rB*
6 D *rA rB*	RLC *rA*, [*rB*]	Rotace *rA* vlevo přes přenos o 16-bit operand adresovaný v paměti *rB*
8 D *rA* x *imm16*	RLC *rA*, *imm16*	Rotace *rA* vlevo přes přenos o *imm16*
B D *rA rB imm16*	RLC *rA*, *rB*, *imm16*	Rotace *rB* vlevo přes přenos o *imm16*, výsledek uložen do *rA*
C D *rA rB offs16*	RLCB *rA*, [*rB*+*offs16*]	Rotace *rA* vlevo přes přenos o 8-bit operand bez znaménka adresovaný v paměti *rB*+*offs16*
E D *rA rB offs16*	RLC *rA*, [*rB*+*offs16*]	Rotace *rA* vlevo přes přenos o 16-bit operand adresovaný v paměti *rB*+*offs16*

Popis

Instrukce provádí rotaci 17 bitů (CF a první operand) vlevo o daný počet pozic.

Operace

```
tempCOUNT ← SRC;
WHILE (tempCOUNT ≠ 0)
DO
tempCF ← MSB(DEST);
DEST ← (DEST * 2) + CF;
CF ← tempCF;
tempCOUNT ← tempCOUNT – 1;
OD; nebo
```

```
DEST ← SRC1;
tempCOUNT ← SRC2;
WHILE (tempCOUNT ≠ 0)
DO
tempCF ← MSB(DEST);
DEST ← (DEST * 2) + CF;
CF ← tempCF;
tempCOUNT ← tempCOUNT – 1;
OD; pro trojoperandový tvar
```

Příznaky

Příznak CF obsahuje bit, který do něj byl nasunut. OF není definován. SF, ZF a AF jsou nastaveny podle pravidel.

ROR – Rotate Right

Opcode	Instrukce	Popis
1 C *rA imm4*	ROR *rA*, *imm4*	Rotace *rA* vpravo o *imm4*
2 C *rA rB*	ROR *rA*, *rB*	Rotace *rA* vpravo o *rB*
3 C *rA* x	ROR *rA*, XAF	Rotace *rA* vpravo o XAF
4 C *rA rB*	RORB *rA*, [*rB*]	Rotace *rA* vpravo o 8-bit operand bez znaménka adresovaný v paměti *rB*
6 C *rA rB*	ROR *rA*, [*rB*]	Rotace *rA* vpravo o 16-bit operand adresovaný v paměti *rB*
8 C *rA* x *imm16*	ROR *rA*, *imm16*	Rotace *rA* vpravo o *imm16*
B C *rA rB imm16*	ROR *rA*, *rB*, *imm16*	Rotace *rB* vpravo o *imm16*, výsledek uložen do *rA*
C C *rA rB offs16*	RORB *rA*, [*rB*+*offs16*]	Rotace *rA* vpravo o 8-bit operand bez znaménka adresovaný v paměti *rB*+*offs16*
E C *rA rB offs16*	ROR *rA*, [*rB*+*offs16*]	Rotace *rA* vpravo o 16-bit operand adresovaný v paměti *rB*+*offs16*

Popis

Instrukce provádí rotaci 16 bitů operandu vpravo o daný počet pozic.

Operace

tempCOUNT ← SRC;
WHILE (tempCOUNT ≠ 0)
DO
CF ← LSB(DEST);
DEST ← (DEST / 2) + (CF * 2^{SIZE});
tempCOUNT ← tempCOUNT – 1;
OD;
ELIHW; nebo

DEST ← SRC1;
tempCOUNT ← SRC2;
WHILE (tempCOUNT ≠ 0)
DO
CF ← LSB(SRC);
DEST ← (DEST / 2) + (CF * 2^{SIZE});
tempCOUNT ← tempCOUNT – 1;
OD;
ELIHW; pro trojoperandový tvar

Příznaky

Příznak CF obsahuje bit, který byl jako poslední přesunut z pozice úplně vpravo, tedy nejvýznamnější bit výsledku. OF není definován. SF, ZF a AF jsou nastaveny podle pravidel.

RRC – Rotate Right through Carry

Opcode	**Instrukce**	**Popis**
1 E *rA imm4*	RRC *rA*, *imm4*	Rotace *rA* vpravo přes přenos o *imm4*
2 E *rA rB*	RRC *rA*, *rB*	Rotace *rA* vpravo přes přenos o *rB*
3 E *rA* x	RRC *rA*, XAF	Rotace *rA* vpravo přes přenos o XAF
4 E *rA rB*	RRCB *rA*, [*rB*]	Rotace *rA* vpravo přes přenos o 8-bit operand bez znaménka adresovaný v paměti *rB*
6 E *rA rB*	RRC *rA*, [*rB*]	Rotace *rA* vpravo přes přenos o 16-bit operand adresovaný v paměti *rB*
8 E *rA* x *imm16*	RRC *rA*, *imm16*	Rotace *rA* vpravo přes přenos o *imm16*
B E *rA rB imm16*	RRC *rA*, *rB*, *imm16*	Rotace *rB* vpravo přes přenos o *imm16*, výsledek uložen do *rA*
C E *rA rB offs16*	RRCB *rA*, [*rB*+*offs16*]	Rotace *rA* vpravo přes přenos o 8-bit operand bez znaménka adresovaný v paměti *rB*+*offs16*
E E *rA rB offs16*	RRC *rA*, [*rB*+*offs16*]	Rotace *rA* vpravo přes přenos o 16-bit operand adresovaný v paměti *rB*+*offs16*

Popis

Instrukce provádí rotaci 17 bitů (CF a první operand) vpravo o daný počet pozic.

Operace

tempCOUNT ← SRC;
WHILE (tempCOUNT ≠ 0)
DO
tempCF ← LSB(DEST);
DEST ← (DEST / 2) + (CF * 2^{SIZE});
CF ← tempCF;
tempCOUNT ← tempCOUNT – 1;
OD; nebo

DEST ← SRC1;
tempCOUNT ← SRC2;
WHILE (tempCOUNT ≠ 0)
DO
tempCF ← LSB(DEST);
DEST ← (DEST / 2) + (CF * 2^{SIZE});
CF ← tempCF;
tempCOUNT ← tempCOUNT – 1;
OD; pro trojoperandový tvar

Příznaky

Příznak CF obsahuje bit, který do něj byl nasunut. OF není definován. SF, ZF a AF jsou nastaveny podle pravidel.

SHL – Shift Left

Opcode	**Instrukce**	**Popis**
1 9 *rA imm4*	SHL *rA*, *imm4*	Posun *rA* vlevo o *imm4*
2 9 *rA rB*	SHL *rA*, *rB*	Posun *rA* vlevo o *rB*
3 9 *rA* x	SHL *rA*, XAF	Posun *rA* vlevo o XAF
4 9 *rA rB*	SHLB *rA*, [*rB*]	Posun *rA* vlevo o 8-bit operand bez znaménka adresovaný v paměti *rB*
6 9 *rA rB*	SHL *rA*, [*rB*]	Posun *rA* vlevo o 16-bit operand adresovaný v paměti *rB*
8 9 *rA* x *imm16*	SHL *rA*, *imm16*	Posun *rA* vlevo o *imm16*
B 9 *rA rB imm16*	SHL *rA*, *rB*, *imm16*	Posun *rB* vlevo o *imm16*, výsledek uložen do *rA*
C 9 *rA rB offs16*	SHLB *rA*, [*rB*+*offs16*]	Posun *rA* vlevo o 8-bit operand bez znaménka adresovaný v paměti *rB*+*offs16*
E 9 *rA rB offs16*	SHL *rA*, [*rB*+*offs16*]	Posun *rA* vlevo o 16-bit operand adresovaný v paměti *rB*+*offs16*

Popis

Instrukce provádí posun bitů operandu vlevo, zprava jsou nasouvány nuly.

Operace

DEST <- DEST << SRC; nebo

DEST <- SRC1 << SRC2; pro trojoperandový tvar

Příznaky

Příznak CF obsahuje poslední vysunutý bit. OF není definován. SF, ZF a AF jsou nastaveny podle pravidel.

SHR – Shift Right

Opcode	Instrukce	Popis
1 A *rA imm4*	SHR *rA*, *imm4*	Logický posun *rA* vpravo o *imm4*
2 A *rA rB*	SHR *rA*, *rB*	Logický posun *rA* vpravo o *rB*
3 A *rA* x	SHR *rA*, XAF	Logický posun *rA* vpravo o XAF
4 A *rA rB*	SHRB *rA*, [*rB*]	Logický posun *rA* vpravo o 8-bit operand bez znaménka adresovaný v paměti *rB*
6 A *rA rB*	SHR *rA*, [*rB*]	Logický posun *rA* vpravo o 16-bit operand adresovaný v paměti *rB*
8 A *rA* x *imm16*	SHR *rA*, *imm16*	Logický posun *rA* vpravo o *imm16*
B A *rA rB imm16*	SHR *rA*, *rB*, *imm16*	Logický posun *rB* vpravo o *imm16*, výsledek uložen do *rA*
C A *rA rB offs16*	SHRB *rA*, [*rB+offs16*]	Logický posun *rA* vpravo o 8-bit operand bez znaménka adresovaný v paměti *rB+offs16*
E A *rA rB offs16*	SHR *rA*, [*rB+offs16*]	Logický posun *rA* vpravo o 16-bit operand adresovaný v paměti *rB+offs16*

Popis

Instrukce provádí logický posun bitů prvního operandu vpravo, zleva jsou nasouvány nuly.

Operace

DEST <- DEST >> SRC; nebo

DEST <- SRC1 >> SRC2; pro trojoperandový tvar

Příznaky

Příznak CF obsahuje poslední vysunutý bit. OF není definován. SF, ZF a AF jsou nastaveny podle pravidel.

ST – Store Register to Memory

Opcode	Instrukce	Popis
7 0 *rA rB*	STB(STS) [*rB*], rA	Uložení 8-bit operandu v rA do paměti na adresu v *rB*
7 1 *rA rB*	ST [*rB*], *rA*	Uložení 16-bit operandu v rA do paměti na adresu v *rB*
F 0 *rA rB offs16*	STB(STS) [*rB+offs16*], rA	Uložení 8-bit operandu v rA do paměti na adresu *rB+offs16*
F 1 *rA rB offs16*	ST [*rB+offs16*], *rA*	Uložení 16-bit operandu v rA do paměti na adresu *rB+offs16*

Popis

Uloží obsah registru rA do paměti na určenou adresu

Šestnáctibitové slovo je uloženo na adresy *mem* a *mem*+1 ve formátu Big Endian. Osmibitové operace jsou ekvivalentní, bez ohledu na typ (byte/short) je vždy uloženo spodních 8 bitů registru na adresu *mem*.

Operace

mem <- rA;

Příznaky

Nezměněny.

STC – Set Carry Flag

Opcode	Instrukce	Popis
0 5 x 1	STC	Nastavení příznaku přenosu

Popis

Instrukce nastavuje příznak CF.

Operace

CF <- 1;

Příznaky

CF nastaveno, ostatní nezměněny.

STI – Set Interrupt Flag

Opcode	Instrukce	Popis
0 5 x 3	STI	Nastavení příznaku povolení přerušení

Popis

Instrukce nastavuje příznak IF. Přerušení jsou povolena.

Operace

IF <- 1;

Příznaky

IF nastaveno, ostatní nezměněny.

SUB – Subtract

Opcode	Instrukce	Popis
1 3 *rA imm4*	SUB *rA*, *imm4*	Odečtení *imm4* od *rA*
2 3 *rA rB*	SUB *rA*, *rB*	Odečtení *rB* od *rA*
3 3 *rA* x	SUB *rA*, XAF	Odečtení XAF od *rA*
4 3 *rA rB*	SUBB *rA*, [*rB*]	Odečtení 8-bit operandu bez znaménka adresovaného v paměti *rB* od *rA*
5 3 *rA rB*	SUBS *rA*, [*rB*]	Odečtení 8-bit operandu se znaménkem adresovaného v paměti *rB* od *rA*
6 3 *rA rB*	SUB *rA*, [*rB*]	Odečtení 16-bit operandu adresovaného v paměti *rB* od *rA*
8 3 *rA* x *imm16*	SUB *rA*, *imm16*	Odečtení *imm16* od *rA*
B 3 *rA rB imm16*	SUB *rA*, *rB*, *imm16*	Odečtení *imm16* od *rB*, výsledek uložen do *rA*
C 3 *rA rB offs16*	SUBB *rA*, [*rB*+*offs16*]	Odečtení 8-bit operandu bez znaménka adresovaného v paměti *rB*+*offs16* od *rA*
D 3 *rA rB offs16*	SUBS *rA*, [*rB*+*offs16*]	Odečtení 8-bit operandu se znaménkem adresovaného v paměti *rB*+*offs16* od *rA*
E 3 *rA rB offs16*	SUB *rA*, [*rB*+*offs16*]	Odečtení 16-bit operandu adresovaného v paměti *rB*+*offs16* od *rA*

Popis

Instrukce provádí znaménkové odčítání druhého operandu od prvního. Výsledek je platný pro znaménkové i neznaménkové operandy, příznaky OF a CF indikují přetečení/přenos znaménkového/neznaménkového výsledku. Příznak SF indikuje znaménko znaménkového výsledku.

Operace

DEST <- DEST - SRC; nebo

DEST <- SRC1 – SRC2; pro trojoperandový tvar

Příznaky

OF, SF, ZF, AF, CF jsou nastaveny podle pravidel.

SBB – Integer Subtraction with Borrow

Opcode	**Instrukce**	**Popis**
1 4 *rA imm4*	SBB *rA*, *imm4*	Odečtení s výpůjčkou *imm4* od *rA*
2 4 *rA rB*	SBB *rA*, *rB*	Odečtení s výpůjčkou *rB* od *rA*
3 4 *rA* x	SBB *rA*, XAF	Odečtení s výpůjčkou XAF od *rA*
4 4 *rA rB*	SBBB *rA*, [*rB*]	Odečtení s výpůjčkou 8-bit operandu bez znaménka adresovaného v paměti *rB* od *rA*
5 4 *rA rB*	SBBS *rA*, [*rB*]	Odečtení s výpůjčkou 8-bit operandu se znaménkem adresovaného v paměti *rB* od *rA*
6 4 *rA rB*	SBB *rA*, [*rB*]	Odečtení s výpůjčkou 16-bit operandu adresovaného v paměti *rB* od *rA*
8 4 *rA* x *imm16*	SBB *rA*, *imm16*	Odečtení s výpůjčkou *imm16* od *rA*
B 4 *rA rB imm16*	SBB *rA*, *rB*, *imm16*	Odečtení s výpůjčkou *imm16* od *rB*, výsledek uložen do *rA*
C 4 *rA rB offs16*	SBBB *rA*, [*rB*+*offs16*]	Odečtení s výpůjčkou 8-bit operandu bez znaménka adresovaného v paměti *rB*+*offs16* od *rA*
D 4 *rA rB offs16*	SBBS *rA*, [*rB*+*offs16*]	Odečtení s výpůjčkou 8-bit operandu se znaménkem adresovaného v paměti *rB*+*offs16* od *rA*
E 4 *rA rB offs16*	SBB *rA*, [*rB*+*offs16*]	Odečtení s výpůjčkou 16-bit operandu adresovaného v paměti *rB*+*offs16* od *rA*

Popis

Instrukce sečte druhý operand s příznakem CF a výsledek následně znaménkově odečte od prvního operandu. Výsledek je platný pro znaménkové i neznaménkové operandy, příznaky OF a CF indikují přetečení/přenos znaménkového/neznaménkového výsledku. Příznak SF indikuje znaménko znaménkového výsledku.

Operace

DEST <- DEST – (SRC + CF); nebo

DEST <- SRC1 – (SRC2 + CF); pro trojoperandový tvar

Příznaky

OF, SF, ZF, AF, CF jsou nastaveny podle pravidel.

XOR – Logical Exclusive OR

Opcode	Instrukce	Popis
1 7 *rA imm4*	XOR *rA*, *imm4*	*imm4* XOR *rA*
2 7 *rA rB*	XOR *rA*, *rB*	*rB* XOR *rA*
3 7 *rA* x	XOR *rA*, XAF	XAF XOR *rA*
4 7 *rA rB*	XORB *rA*, [*rB*]	8-bit operand bez znaménka adresovaný v paměti *rB* XOR *rA*
5 7 *rA rB*	XORS *rA*, [*rB*]	8-bit operand se znaménkem adresovaný v paměti *rB* XOR *rA*
6 7 *rA rB*	XOR *rA*, [*rB*]	16-bit operand adresovaný v paměti *rB* XOR *rA*
8 7 *rA* x *imm16*	XOR *rA*, *imm16*	*imm16* XOR *rA*
B 7 *rA rB imm16*	XOR *rA*, *rB*, *imm16*	*imm16* XOR *rB*, výsledek uložen do *rA*
C 7 *rA rB offs16*	XORB *rA*, [*rB*+*offs16*]	8-bit operand bez znaménka adresovaný v paměti *rB*+*offs16* XOR *rA*
D 7 *rA rB offs16*	XORS *rA*, [*rB*+*offs16*]	8-bit operand se znaménkem adresovaný v paměti *rB*+*offs16* XOR *rA*
E 7 *rA rB offs16*	XOR *rA*, [*rB*+*offs16*]	16-bit operand adresovaný v paměti *rB*+*offs16* XOR *rA*

Popis

Instrukce provádí operaci XOR mezi jednotlivými bity operandů.

Operace

DEST <- DEST XOR SRC; nebo

DEST <- SRC1 XOR SRC2; pro trojoperandový tvar

Příznaky

OF a CF jsou vynulovány, SF, ZF a AF jsou nastaveny podle pravidel.