Die Architektur des fiktiven Prozessors P/F (Kurzbeschreibung)

Sinn und Zweck
P/F wurde mit dem Ziel entwickelt, eine moderne Prozessorarchitektur bereitzustellen, die zu Demonstrations- und Lehrzwecken sowie zum Dokumentieren elementarer Programmabläufe (z. B. von Testprogrammen, Systemkernen und E-A-Funktionen) brauchbar ist. P/F ist aber keine Spielzeug-Architektur. Sie ist einerseits viel einfacher als Architekturen wie beispielsweise IA-32, IA-64 oder PowerPC, andererseits aber frei von den Einschränkungen, die die typischen kleinen Mikrocontroller dem Programmierer auferlegen. P/F wurde ausdrücklich als konservative (herkömmliche) Architektur angelegt.

Anmerkungen:

1. die Architekturdefinition ist freizügig erweiterbar (mehr Register, weitere Operationen usw.),
2. die P/F-Befehle lassen sich weitgehend 1:1 auf die Befehlslisten der gängigen Prozessoren abbilden, so daß P/F-Quelltexte mit praktisch jedem beliebigen Macro-Assembler in “echte” Maschinenprogramme umgesetzt werden können (P/F-Architekturdefinition => Macro-Definition),
3. im Sinne verschiedener Lernziele ist es möglich, entsprechend “abgemischte” Befehlssätze bereitzustellen (wobei die jeweils unnötigen Merkmale entfallen).

Die folgende Kurzbeschreibung entspricht dem Ausgabestand 1.4 des P/F-Simulators. Diese Befehlsliste steht für ein mitttleres Niveau der Kompliziertheit, wie es für viele Programmierübungen vollauf ausreicht.

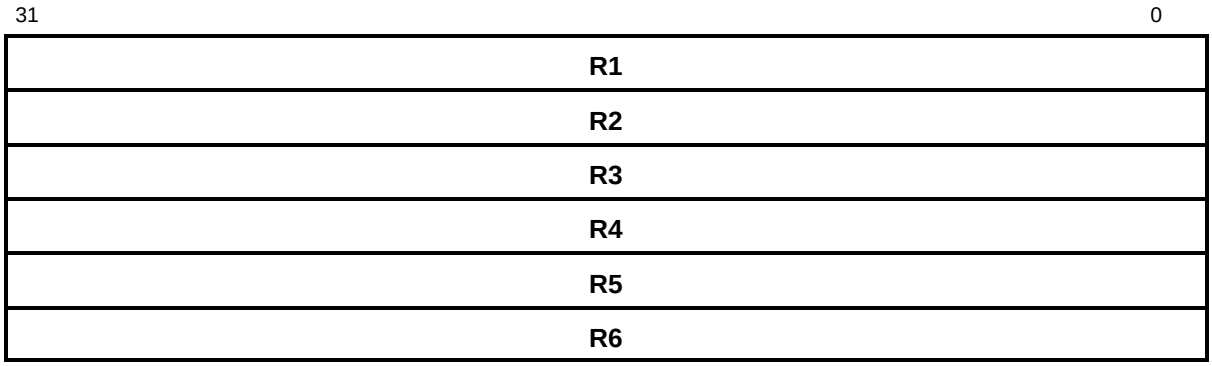
Technische Daten und Merkmale im Überblick:

- Verarbeitungsbreite: 32 Bits,
- Architekturprinzip: RISC; Load-Store-Organisation. 2-Operanden-Befehle,
- Befehlslänge: in diesem Ausgabestand nicht festgelegt. Bei ausschließlichem Bezug auf symbolische Adressierung und Assemblernotation bedeutungslos.
- Registersatz: 6 Universalregister R1...R6, 1 Stackpointer SP, ein Befehlszähler PC, ein Zustandsregister ST,
- Adressierung und Bit-Indizierung: Rechtsadressierung und Rechtsindizierung (Little Endian),
- Speicheradressierung: linearer Adreßraum, 32-Bit-Adresse. Keine Adreßumsetzung.
- E-A-Adressierung: linearer Adreßraum, 32-Bit-Adresse. Keine Adreßumsetzung.
- Datenstrukturen werden nur an ihren jeweiligen integralen Adressen gespeichert.

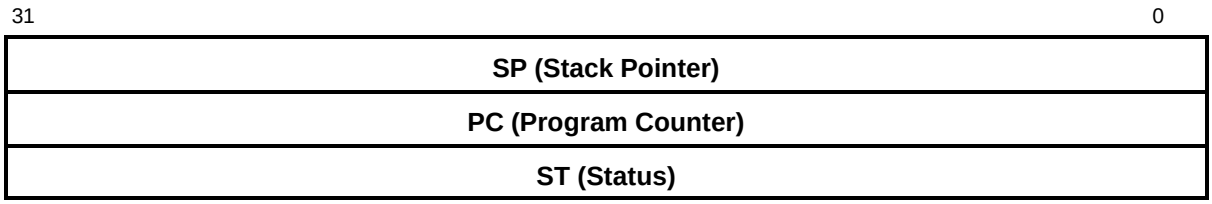
- Programmstart nach Rücksetzen: von Adresse 0 an.
- Interrupts: in diesem Ausgabestand nicht vorgesehen.
- Expansionsrichtung des Stack: von höheren zu niederen Adressen. Push = Predecrement, Pop = Postincrement.

Registersatz

Universalregister:

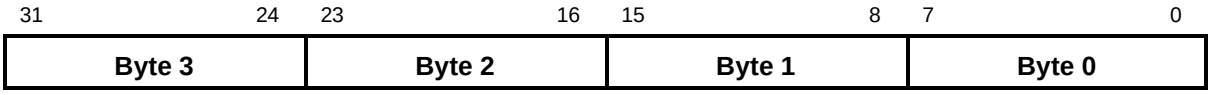


Sonderregister:

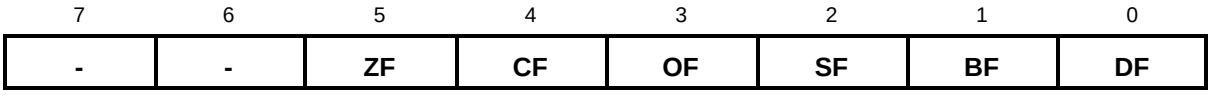


Ab Ausgabestand 1.6 kommt ein Frame bzw. Base Pointer (BP) hinzu.

Einzelheit: Anordnung der Bytes im 32-Bit-Wort:



Einzelheit: die Bits des Zustandsregisters (ST):



Ungenutzte Bits (-) sind fest mit Null belegt (Überladen ist wirkungslos). Die Bytes 3, 2, 1 sind reserviert.

Flagbits

Mnemonic	Bezeichnung	Bedeutung
ZF	Zero Flag	Ergebnis = Null
CF	Carry Flag	Ausgangsübertrag
OF	Overflow Flag	Überlauf = Augangsübertrag ⊕ Übertrag in höchstwertige Stelle
SF	Sign Flag	Vorzeichen (= höchstwertiges Bit des Ergebnisses)
BF	BCD Carry Flag	Übertrag von Bitposition 3 in Bitposition 4
DF	Dummy Flag	reserviert. Programmseitig stellbar

Transportbefehle

Mnemonic	Bezeichnung	Wirkung
LDI Rn,Imm32	Load Immediate (Laden Direktwert)	<Rn> := Imm32
LD Rn, Daddr	Load (Laden). Daddr ist 32-Bit-Wortadresse	<Rn> := <Daddr>
LDR Rn,Ra	Load Indirect (Laden indirekt)	<Rn> := <<Ra>>
ST Rn, Daddr	Store (Speichern). Daddr ist 32-Bit-Wortadresse	<Daddr> := <Rn>
STR Rn,Ra	Store Indirect (Speichern indirekt)	<<Ra>> : = <Rn>
MOV Rd, Rs	Move (Transportieren)	<Rd> := <Rs>
LDSR Rn,SR	Load from Special Register (Laden von Sonderregister)*)	<Rn> =: <Sr>
STSR Rn,SR	Store to Special Register (Speichern nach Sonderregister)*)	<Sr> := <Rn>

Flagbits: werden nicht verändert. *): zulässige Mnemonics: PC, SP, ST

Arithmetische und logische Befehle

Mnemonic	Bezeichnung	Wirkung	Flagbits				
			BF	SF	OF	CF	ZF
ADD Rx, Ry	Add (Addieren)	<Rx> := <Rx> + <Ry>	j	j	j	j	j
SUB Rx,Ry	Subtract (Subtrahieren)	<Rx> := <Rx> - <Ry>	j	j	j	j	j
ADDC Rx,Ry	Add with Carry (Addieren mit Eingangsübertrag)	<Rx> := <Rx> + <Ry> + CF	j	j	j	j	j
SUBC Rx,Ry	Subtract with Carry (Subtrahieren mit Eingangsübertrag)	<Rx> := <Rx> - <Ry> + CF	j	j	j	j	j
CPL Rn	Two's Complement (Zweierkomplement)	<Rn> := 2^n - <Rn>	j	j	j	j	j
INC Rn,Imm32	Increment (Erhöhen)	<Rn> := <Rn> + Imm32	j	j	j	j	j
DEC Rn,Imm32	Decrement (Vermindern)	<Rn> := <Rn> - Imm32	j	j	j	j	j

Mnemonic	Bezeichnung	Wirkung	Flagbits				
			BF	SF	OF	CF	ZF
CMP Rx,Ry	Compare (Vergleichen)	<Rx> - <Ry> (kein Ergebnis)	j	j	j	j	j
INV Rn	Invert (Negation)	<Rn> := /<Rn>	n	j	n	n	j
AND Rx,Ry	AND (UND-Verknüpfung)	<Rx> := <Rx> /\ <Ry>	n	j	n	n	j
OR Rx,Ry	OR (ODER-Verknüpfung)	<Rx> := <Rx> \/ <Ry>	n	j	n	n	j
XOR Rx,Ry	XOR (Antivalenz-verknüpfung)	<Rx> := <Rx> ⊕ <Ry>	n	j	n	n	j

j: Flagbit wird gemäß Ergebnis gestellt, n: Flagbit wird nicht verändert

Verschieben und Rotieren

Der 1. Operand wird gemäß der Angabe des 2. Operanden verschoben (2. Operand = 0: kein Verschieben, 2. Operand = 1: Verschieben um 1 Bit usw.). Bei SHIFT werden freiwerdende Bitpositionen mit Nullen aufgefüllt, bei ROT laufen die herausgeschobenen Bits in die freiwerdenden ein.

Shift Right Arithmetically: die freiwerdenden Bits werden mit dem Vorzeichenbit aufgefüllt.

Rotate Extended: die Rotation wird über CF geführt (der aktuelle Inhalt von CF wird in die jeweils erste Bitposition übernommen und das jeweils herausgeschobene Bit wiederum in CF). Rotation nur um ein Bit.

Mnemonic	Bezeichnung	Flagbits				
		BF	SF	OF	CF*)	ZF
SHL Rn,Imm8	Shift Left	n	j	n	j	j
SHR Rn,Imm8	Shift Right	n	j	n	j	j
ROTL Rn,Imm8	Rotate Left	n	j	n	j	j
ROTR Rn,Imm8	Rotate Right	n	j	n	j	j
SHLR Rn,Rc	Shift Left according to Register	n	j	n	j	j
SHRR Rn,Rc	Shift Right according to Register	n	j	n	j	j
ROTLR Rn,Rc	Rotate Left according to Register	n	j	n	j	j
ROTRR Rn,Rc	Rotate Right according to Register	n	j	n	j	j
SRA Rn,Imm8	Shift Right arithmetically	n	j	n	j	j
SRAR Rn,Rc	Shift Right arithmetically according to Register	n	j	n	j	j
RTLX Rn	Rotate Left Extended	n	j	n	j	j
RTRX Rn	Rotate Right Extended	n	j	n	j	j

j: Flagbit wird gemäß Ergebnis gestellt, n: Flagbit wird nicht verändert.

*): CF enthält das zuletzt herausgeschobene Bit

Allgemeine Programmsteuerbefehle

Expansionsrichtung des Stack: von höheren zu niederen Adressen. Push = Predecrement, Pop = Postincrement. Flagbits werden nicht verändert (aber bei entsprechenden POP-Befehlen gemäß Stackinhalt gestellt).

Mnemonic	Bezeichnung	Wirkung
PUSH Rn	Push Register onto Stack	SP vermindern, danach Register n (eines der Register R1...R6) auf Stack legen
POP Rn	Pop Register from Stack	Stackbelegung in Register n transportieren, danach SP erhöhen
PUSHSTA	Push Status Reg onto Stack	SP vermindern, danach Statusregister auf Stack
POPSTA	Pop Status Reg from Stack	Statusregister vom Stack nehmen, danach SP erhöhen
PUSHA	Push all Registers onto Stack	SP vermindern, danach Register auf Stack. Reihenfolge: STA, R6, R5...R1
POPA	Pop all Registers from Stack	Register von Stack nehmen. Reihenfolge: R1, R2...R6, STA. Danach SP erhöhen
CALL Daddr	Call Subroutine	SP vermindern, danach Befehlszähler auf Stack legen. Gemäß Daddr verzweigen
SBRA Ra	Subroutine Branch Indirect	SP vermindern, danach Befehlszähler auf Stack legen. Mit Registerinhalt von Ra verzweigen
RET	Return from Subroutine	Befehlszähler vom Stack aus laden, danach SP erhöhen
JMP Daddr	Jump (unconditional)	gemäß Daddr verzweigen. <PC> := Daddr
BRA Ra	Branch Indirect	mit Registerinhalt von Ra verzweigen. <PC> := <Ra>
NOP	No Operation	keine Wirkung
HALT *Message*	Halt Instruction Execution	Ende der Befehlsausführung. Rücksetzen erforderlich. *Message*-Text wird angezeigt

Bedingte Verzweigungen

Mnemonic	Bezeichnung	Wirkung	Verzweigungs-bedingung
JPZ, JPE	Jump if Zero, Jump if Equal	Verzweigen, wenn Null bzw. Gleichheit	ZF = 1
JPNZ, JPNE	Jump if Not Zero, Jump if Not Equal	Verzweigen, wenn nicht Null bzw. Ungleichheit	ZF = 0
JPC	Jump if Carry	Verzweigen, wenn Ausgangsübertrag	CF = 1
JPNC	Jump if No Carry	Verzweigen, wenn kein Ausgangsübertrag	CF = 0
JPO	Jump if Overflow	Verzweigen, wenn Überlauf	OF = 1
JPNO	Jump if No Overflow	Verzweigen, wenn kein Überlauf	OF = 0
JPP	Jump if Positive	Verzweigen, wenn negativ	SF = 1
JPN	Jump if Negative	Verzweigen, wenn positiv	SF = 0
JPBC	Jump if BCD Carry	Verzweigen, wenn BCD-Flag gesetzt	BF = 1
JPNBC	Jump if No BCD Carry	Verzweigen, wenn BCD-Flag gelöscht	BF = 0
JPD	Jump if Dummy Flag Set	Verzweigen, wenn Dummy-Flag gesetzt	DF = 1
JPND	Jump if Dummy Flag Cleared	Verzweigen, wenn Dummy-Flag gelöscht	DF = 0
Verzweigungsbedingungen für vorzeichenlose (natürliche) Binärzahlen			
JPB	Jump if Below	Verzweigen, wenn A < B	ZF = 0 & CF = 0
JPA	Jump if Above	Verzweigen, wenn A > B	ZF = 0 & CF = 1
JPBE	Jump if Below or Equal	Verzweigen, wenn A ≤ B	ZF = 1 v CF = 0
JPAE	Jump if Above or Equal	Verzweigen, wenn A ≥ B	ZF = 1 v CF = 1
Verzweigungsbedingungen für vorzeichenbehaftete (ganze) Binärzahlen			
JPL	Jump if Less	Verzweigen, wenn A < B	OF ⊕ SF = 1
JPG	Jump if Greater	Verzweigen, wenn A > B	OF ⊕ SF = 0
JPNG, JPLE	Jump if Not Greater (= Less or Equal)	Verzweigen, wenn A ≤ B	ZF v (OF ⊕ SF) = 1
JPNL, JPGE	Jump if Not Less (= Greater or Equal)	Verzweigen, wenn A ≥ B	ZF = 1 v (OF ⊕ SF) = 0

Bitbefehle

Im ersten Operanden wird das Bit ausgewählt, das der 2. Operand adressiert. 2. Operand = 0: Bit 0, 2. Operand = 1: Bit 1 usw. Mit Ausnahme von ZF bei BIT und BITR werden die Flagbits nicht verändert.

Mnemonic	Bezeichnung	Wirkung
BIT Rn,Imm8	Test Bit	gemäß der Belegung des adressierten Bits wird das Flagbit ZF gestellt: ZF = 0, wenn Bit = 1, ZF = 1, wenn Bit = 0
SET Rn,Imm8	Set Bit	das adressierte Bit wird gesetzt (zu 1)
CLR Rn,Imm8	Clear Bit	das adressierte Bit wird gelöscht (zu 0)
BITR Rn,Rs	Test Bit according to Register	gemäß der Belegung des adressierten Bits wird das Flagbit ZF gestellt: ZF = 0, wenn Bit = 1, ZF = 1, wenn Bit = 0
SETR Rn,Rs	Set Bit according to Register	das adressierte Bit wird gesetzt (zu 1)
CLRR Rn,Rs	Clear Bit according to Register	das adressierte Bit wird gelöscht (zu 0)
BZF Rn, Imm8	Copy Inverted ZF to Bit	die invertierte Belegung des Zero Flag wird in das adressierte Bit kopiert
BZFR Rn, Rs	Copy Inverted ZF to Bit according to Register	die invertierte Belegung des Zero Flag wird in das adressierte Bit kopiert

Ein- und Ausgabe

Mnemonic	Bezeichnung	Wirkung
OUT Rn,Imm32	Output from Register	Port gemäß Portadresse (Imm32) := <Rn>
IN Rn,Imm32	Input to Register	<Rn> := Port gemäß Portadresse (Imm32)

Flagbits: werden nicht verändert

Das Simulationsprogramm PFSIM stellt verschiedenartige Übungsperipherie zur Verfügung (u. a. ein Schalterfeld und einen Textbildschirm).

Fehlersuchen

Mnemonic	Bezeichnung	Wirkung
CHECKPOINT *Message* oder CHK *Message*	Checkpoint Display (Anzeigen einer Textmitteilung)	*Message*-Text wird angezeigt; Programm läuft weiter
BREAKPOINT oder BRK	Breakpoint-Anzeige	Breakpoint-Anzeige mit Nummer der Programmzeile. Stop. Fortsetzung durch manuelle Bestätigung
PRAGMA *Message*	Beeinflussung der Entwicklungs- bzw. Ablaufumgebung	*Message* veranlaßt verschiedenartige Wirkungen in der Entwicklungs- bzw. Ablaufumgebung. Erklärung in Beschreibung PFSIM

Flagbits: werden nicht verändert