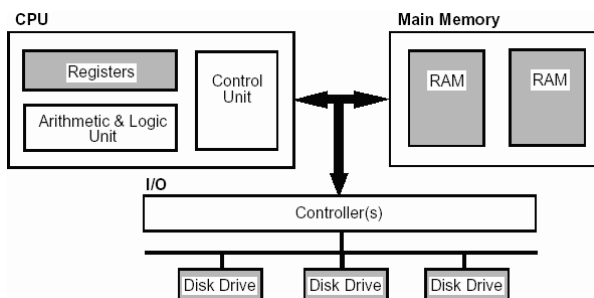


POT (T.Mainzer)

Přehled: CPU - paměť

CU-ALU-Registers ↔ Memory (Address bus, data bus, control bus)



Externí paměť – pomalá → registry, cache

Procesor H8S

Registry – $8 \times 32\text{bit ER}_n (=16 \times 16\text{bit En}+\text{Rn}, 8 \times 16\text{bit}+16 \times 8\text{bit En}+\text{RnH}+\text{RnL})$

31 _____ ERx _____ 0
15 _____ Ex _____ 0 15 _____ Rx _____ 0
7 _____ RxH _____ 0 7 _____ RxL _____ 0

E0	R0H	R0L	ER0
E1	R1H	R1L	ER1
...	
...	
...	
...	
E7	R7H	R7L	ER7

ER7 = speciální fce, ukazatel zásobníku

Instrukce procesoru

čtení z paměti, dekódování, provádění

mnemotechnický zápis instrukcí – assembler

Aplikační progr. (database, simulatory) – Prog.jazyky (Java,C) – **Assembler/Stroj.kod** – logické obvody – El.signaly

Základní operace s registry Přesuny mezi registry:

MOV.B zdroj, cíl byte – přesun (8bitů = B)

MOV.W zdroj, cíl word - přesun (16bitů = W) , e.g. mov.w R1,R0 ;přesun obsahu reg.R1 do R0

MOV.L zdroj, cíl long - přesun (32bitů = L)

Uložení konstanty do registru

mov.w #konst:16, R0 (#=konstanta, :16 = word) (i adresa do paměti je konstanta)

Uložení hodnoty z paměti do registru (pozn. 16 a 32bit. Císla musí být zarovnaná na sudé adresy)

mov.w @1234:16, R0 ...obsah adresy 1234 do reg.R0, absolutní adresa

První program

```
.h8300s
.data ;datová část
var1: .word 0x1778 ;návěští – jméno proměnné, proměnná, s definovanou počáteční hodnotou
var2: .word 0x5678 ;návěští – jméno proměnné, proměnná, s definovanou počáteční hodnotou
var3: .space 2      ; 2 byte (=word) , neinicialisovane
      .space 100    ; 100 bytu pro zásobník , neinicialisovano
stack:
.text ;programová část
.global _start
_start:
mov.l #stack,ER7 ; hodnota stack do SP (tj. Adresa #stack)
mov.w @var1,R0 ;načtení hodnoty z paměťového prostoru, kam ukazuje „var1“
add.w #7654:16,R0 ;součet konstanta+R0 -> R0
mov.w R0,@var3 ;ulození výsledku součtu do var2
lab1: jmp @lab1 ;nekonečná smyčka
.end
```

Další instrukce

Sčítání:

ADD #konst,Reg ; konst+reg -> reg

ADD reg1,reg2 ; reg1+reg2 -> reg2

Inkrementace/dekrementace (INC, DEC):

INC reg ;reg=reg+1 (DEC reg pro dekrementaci)

INC n,reg ;reg=reg+n (n=#1,#2)

Porovnání (reg2-reg1 -> použití pro porovnání reg2 cc reg1, kde cc je podmínka skoku)

CMP #konst,reg ; porovnání, dle výsledku operace reg-#konst nastaví příznakové bity

CMP reg1,reg2 ; porovnání, dle výsledku operace reg2-reg1 nastaví příznakové bity

Skoky - nepodmíněné

JMP @návěští ;skok na adresu

JMP @reg ;skok na adresu danou obsahem registru

Skok – podmíněný (cc=podmínka založená na příznacích, viz instr.soubor)

Bcc návěští ;podmíněný skok na návěští (návěští je bez @ neb se jedná o relativní skok)

Skoky – volání podprogramu

JSR @návěští ;skok do podprogramu (volání fce)

RTS ;návrat z podprogramu (return)

Práce se zásobníkem

PUSH reg ;vlození do zásobníku (PUSH.W a PUSH.L) (=DEC ER7, MOV reg,@ER7)

POP reg ;vybrání ze zásobníku (POP.W, POP.L) (mov @ER7,reg, INC ER7)

Příklad:

Prohození (swap) hodnot reg. R0 a R1

Přes registr	Přes paměť	Přes zásobník	Přes zásobník 2	Xor swap
mov.w R0,R2 mov.w R1,R0 mov.w R2,R1	mov.w R0,@var mov.w R1,R0 mov.w @var,R1	Push.w R0 Push.w R1 pop.w R0 pop.w R1	Push.w R0 mov.w R1,R0 pop.w R1	xor.w r1,r0 xor.w r0,r1 xor.w r1,r0

16bit sčítání R0+R1 -> R1

Přímé, 16bit	Po 8 bitech, chybné bez přenosu	Po 8 bitech, s přenosem	Po 8 bitech s přenosem, ADDX
add.w r0,r1	add.b r0l,r1l add.b r0.h,r1h	Add.b r0l,r1l bcc skip ;není-li přenos, skoc inc r1h ;připocti přenos skip: add r0h,r1h	Add.b r0l,r1l addx r0h,r1h

Převod hexdec.číslice ascii ↔ číslo

Převod šestnáctkové číslice uložené v proměnné jako znak (0..9,A..F příp. a..f)	Převod čísla 0-15 na jeho znakový ekvivalent (0-F)	Tabulka Znaků ASCII
.h8300s .data ;datova cast char1: .word 0x0066 ;znak 'f' .text ; .global _start _start: mov.w @char1,R0 cmp.b #0x3A:8,R0l ;'9'+1 bgt znak:16 ;jump if >='9' cislo: sub.w #0x30:16,R0 ;-'0' jmp @lab1 znak: cmp.b #0x46:8,R0l ;'F' bgt maly:16 ;>'F' velky: sub.w #0x4b:16,R0 ;-'A'+10 jmp @lab1 maly: sub.w #0x6b:16,R0 lab1: jmp @lab1 .end	.h8300s .data ;datova cast char1: .word 0x000F ;cislo 15 .text ; .global _start _start: mov.w @char1,R0 cmp.b #0x09:8,R0l bgt znak:16 ;skok kdyz R0l > 9 cislo: add.w #0x30:16,R0 ;+'0' jmp @lab1 znak: add.w #0x37:16,R0 ;+'A'-10 lab1: jmp @lab1 .end	'0' 0x30 '1' 0x31 .. '9' 0x39 .. 'A' 0x41 'B' 0x42 .. 'F' 0x46 .. 'a' 0x61 .. 'f' 0x66

Další instrukce

SHAL/SHAR reg ;aritmeticky shift vlevo/vpravo (=zachování znaménka) (=fce *2, /2 dopl.kod)

SHLL/SHLR reg ;logický shift (neznaménkový) (=fce *2, /2 neznaménkové)

ROTL/ROTR ;rotace (pozn. shifty i rotace „vysunují“ bit do carry)

ROTXL/ROTXR reg ;rotace přes carry (carry je jako n+1 bit)

AND,OR,XOR,NOT ;logické operace

Podmíněné skoky

Instrukce	Význam (skok když...)	Podmínka (skok když...)
BRA(BT)	Always (true)	Always
BRN(BF)	Never (false)	Never
BHI	High	$C \vee Z = 0$
BLS	Low or same	$C \vee Z = 1$
BCC(BHS)	Carry clear (high or same)	$C = 0$
BCS(BLO)	Carry set (low)	$C = 1$
BNE	Not equal	$Z = 0$
BEQ	Equal	$Z = 1$
BVC	Overflow clear	$V = 0$
BVS	Overflow set	$V = 1$
BPL	Plus	$N = 0$
BMI	Minus	$N = 1$
BGE	Greater or equal	$N \oplus V = 0$
BLT	Less than	$N \oplus V = 1$
BGT	Greater than	$Z \vee (N \oplus V) = 0$
BLE	Less or equal	$Z \vee (N \oplus V) = 1$

Příklady užití:

cmp #58,R0 bgt jevetsi ;skok, je-li R0>58	Rotl R1,1 ;rotace R1 vlevo, nejvyšší bit R1 do C bcs cnula ;skok je-li C(array)=1
cmp #32,R0 beq jestejne ;skok, je-li R0=32	bld 7,R0L ;7-mý bit R0L do C(array) (jen byte) bcc cnula ;skok je-li C(array)=0

Tabulky instrukcí:

Mnemonic		Size	Addressing Mode and Instruction Length (Bytes)								Operation	Condition Code						No. of States*1	
			#xx	Rn	@ERn	@(d,ERn)	@-ERn/@ERn+	@aa	@(d,PC)	@@aa			I	H	N	Z	V	C	Normal
MOV	MOV.B #xx:8,Rd	B	2								#xx:8→Rd8	—	—	↓	↓	0	—	1	
	MOV.B Rs,Rd	B		2							Rs8→Rd8	—	—	↓	↓	0	—	1	
	MOV.B @ERs,Rd	B			2						@ERs→Rd8	—	—	↓	↓	0	—	2	
	MOV.B @(d:16, ERs), Rd	B				4					@(d:16,ERs)→Rd8	—	—	↓	↓	0	—	3	
	MOV.B @(d:32,ERs),Rd	B				8					@(d:32,ERs)→Rd8	—	—	↓	↓	0	—	5	

Význam jednotlivých polí: Název instrukce – formát instrukce – typ operandů (B/W/L) – kolik byte má instrukce s danými operandy, operace (co dělá instrukce), ovlivnění příznakových bitů (= beze změny, | = ovlivněno dle výsledku, 0/1 = nastaveno), počet taktů na provedení instrukce

pozn. podmíněné skoky mají různý počet taktů v závislosti na tom jestli je skok proveden nebo ne

pozn. kolik byte = délka programu (-> obsazení paměti), počet taktů = jak dlouho trvá provedení instrukce (-> rychlost programu).

Délka (pod)programu, doba trvání:

START: mov #5,R0 ; pocitadlo NEXT: dec R0L ; pocitadlo - 1 bne NEXT ; opakovani smycky rts	4Byte, 2Takty (= mov.W #xx:16,Rd) 2Byte, 1Takt (dec.b Rd) 2byte, 3(skok) nebo 2 (pokračování) Takty (bne d:8) 2byte, 4 Takty
---	---

Délka: 4+2+2+2byte = 10byte

Doba: 2 + 5*1 + (4*3+1*2) + 4 = 25Taktů

Funkce – počítání jedničkových bitů

<pre> .h8300s ; FCE - prenos parametru pres registry ; vstup: R1 - cislo ; vystup: R0H - pocet jednicek ; modif.: R0,R1 SUB: mov.b #16,R0L ; pocitadlo rotaci xor.b R0H,R0H ; citac jednicek, snulování NEXT: rotr R1 ; rotace vpravo addx.b #0,R0H ; pricte Carry do poctu jednicek dec R0L ; pocitadlo - 1 bne NEXT ; dalsi rotace rts .data VAR1: .word 0xff ;vstupni parametr VAR2: .word 0x13 RES1: .byte 0 ;vystupni hodnota RES2: .byte 0 .space 100 STCK: .text .global _start _start: mov.l #STCK,ER7 ; inicializace zasobniku ; priklad volani podprogramu mov.w @VAR1,R1 ; vstupni parametr JSR @SUB ; volani podprogramu mov.b R0H,@RES1 ; ulozi vystup do pameti end: bra end ; nekonecna smycka .end </pre>	<pre> .h8300s ; FCE: prenos parametru odkazem ; vstup: ER2 - adresa cisla (word) ; vystup: ER3 - adresa vysledku (byte) ; modif.: R0,R1,ER2,ER3 SUB: mov.w @ER2,R1 ; vybrani parametru delky word mov.b #16,R0L ; pocitadlo rotaci xor.b R0H,R0H ; citac jednicek NEXT: rotr R1 ; rotace vpravo addx.b #0,R0H ; pricte Carry do poctu jednicek dec R0L ; pocitadlo - 1 bne NEXT ; dalsi rotace mov.b R0H,@ER3; ulozeni vysledku delky byte rts .data VAR1: .word 0xff VAR2: .word 0x13 RES1: .byte 0 RES2: .byte 0 .space 100 STCK: .text .global _start _start: mov.l #STCK,ER7 ; inicializace zasobniku ;priklad volani podprogramu: mov.l #VAR1,ER2 ; vstupni parametr (adresa) mov.l #RES1,ER3 ; vystupni parametr (adresa) JSR @SUB ; volani podprogramu end: bra end ; nekonecna smycka .end </pre>
<pre> .h8300s ; SUB: prenos parametru pres zasobnik 1 ; vstup: word - cislo v zasobniku ; vystup: word - pocet (pouze horni byte) ; modif.: R0,R1,ER2 SUB: pop.l ER2 ; ze zas. adresu navratu ->ER2 pop.w R1 ; vybere parametr fce mov.b #16,R0L ; pocitadlo rotaci xor.b R0H,R0H ; citac jednicek NEXT: rotr R1 ; rotace vpravo addx.b #0,R0H ; pricte Carry do poctu jednicek dec R0L ; pocitadlo - 1 bne NEXT ; dalsi rotace push.w R0 ; ulozi vysledek push.l ER2 ; obnovi adresa navratu rts .data VAR1: .word 0xff VAR2: .word 0x13 RES1: .byte 0 RES2: .byte 0 .space 100 STCK: .text .global _start _start: mov.l #STCK,ER7 ; inicializace zasobniku ; příklad volani podprogramu mov.w @VAR1,R1 ; vstupni parametr push.w R1 ; ulozi do zasobniku JSR @SUB ; podprogram pop.w R1 ; vystup ze zasobniku mov.b R1H,@RES1 ; ulozi do pameti end: bra end ; nekonecna smycka .end </pre>	<pre> .h8300s ; SUB: prenos parametru pres zasobnik 2 ; vstup: word - cislo v zasobniku ; vystup: word - pocet (pouze horni byte) ; modif.: R0,R1,ER2 SUB: mov.l ER7,ER2 ; pozice zasobniku add.l 4,ER2 ; ukazatel na parametr mov.w @ER2,R1 ; precteni parametru ;mov.w @(4,ER7),R1 ; altern.predchozich 3radku mov.b #16,R0L ; pocitadlo rotaci xor.b R0H,R0H ; citac jednicek NEXT: rotr R1 ; rotace vpravo addx.b #0,R0H ; pricte Carry do poctu jednicek dec R0L ; pocitadlo - 1 bne NEXT ; dalsi rotace mov.w R0,@ER2 ; ulozi vysledek rts .data VAR1: .word 0xff VAR2: .word 0x13 RES1: .byte 0 RES2: .byte 0 .space 100 STCK: .text .global _start _start: mov.l #STCK,ER7 ; inicializace zasobniku ; příklad volani podprogramu mov.w @VAR1,R1 ; vstupni parametr push.w R1 ; ulozi do zasobniku JSR @SUB ; podprogram pop.w R1 ; vystup ze zasobniku mov.b R1H,@RES1 ; ulozi do pameti end: bra end ; nekonecna smycka .end </pre>

<pre> .altmacro ; macro zjistí počet jedniček ve wordu ;macro ONES VAR,RESULT LOCAL NEXT ; lokální navesti mov.w @\VAR,R1 ; vybrání parametru delky word mov.b #16,R0L ; pocítadlo rotací xor.b R0H,R0H ; citací jedniček NEXT: rotr R1 ; rotace vpravo addx.b #0,R0H ; přičte Carry do počtu jedniček dec R0L ; pocítadlo - 1 bne NEXT ; další rotace mov.b R0H,@\RESULT ; uložení výsledku delky byte .endm .data VAR1: .word 0xff VAR2: .word 0x13 RES1: .byte 0 RES2: .byte 0 .space 100 STCK: .text .global _start _start: mov.l #STCK,ER7 ; inicializace zásobníku ONES VAR1,RES1 ONES VAR2,RES2 end: bra end ; nekonečná smyčka .end </pre>	<pre> .altmacro ;macro INC2M VAR ;zvysit VAR o 2 inc \VAR inc \VAR .endm ;macro INC4M VAR ;zvysit VAR o 4 INC2M \VAR INC2M \VAR .endm INC2F: ;funkce - zvysit R0 o 2 inc R0 inc R0 rts INC4F: jsr @INC2F ;fce - zvysit R0 o 4 jsr @INC2F rts .text .global _start _start: mov.l #STCK,ER7 ; inicializace zásobníku mov 12,R0 mov 12,R1 INC4M R0 ;R0+4 - rozvin makra = 4*INC R0 INC4M R0 ;R0+4 - rozvin makra = 4*INC R0 jsr @INC4F ;R0+4, volání fce jsr @INC4F ;R0+4, další volání fce INC4M R1 ;R1+4 - rozvin makra = 4*INC R1 mov R1,R0 ;presun parametru R1->R0 jsr @INC4F ;volání FCE (R0+4) mov R0,R1 ;presun parametru -> R1+4 end: bra end ; nekonečná smyčka .end </pre>
--	--

Adresní módy

Symbol.	příklad	Adresní mód	Pozn.
Rn	mov r0,r1	Přímý registrový	Parametrem je registr (RxH,RxL,Rx,Ex,ERx)
@ERn	mov r0,@er1	Nepřímý registrový	Parametr je přečten z adresy dané registrem
@(d,ERn)	mov r0,@(4,er1)	Nepřímý registrový s offsetem	Parametr je přečten z adresy dané součtem registru+offsetu. Offset = konstanta
@ERn+	mov r0,@er1+	Nepřímý registrový s post incrementem	Parametr je přečten z adresy dané registrem, registr(tj.ukazatel) je po té inkrementován
@-ERn	mov r0,@-er1	Nepřímý registrový s pre decrementem	Parametr je přečten z adresy dané registrem, ale registr je ještě předtím dekrementován
@aa	mov r0,@label	Absolutní adresa	Parametr je přečten z absolutní adresy (tj. adresa=konstanta, typicky návěští)
#x	mov r0,#1234	Přímý (konstanta)	Parametrem je konstanta (číslo)
@(d,PC)	beq @loop	Relativní k PC	Adresa je dána relativně k Program counteru. Užité v instrukcích podmíněného skoku – tj. PC+d, obvykle počítá překladač
@@aa	jsr @@4	Paměťový nepřímý	Adresa je přečtena z (konstantní) adresy. Užité ve spec.případech skoku JMP a JSR (tabulka vektorů přerušení)

Volání systémových fcí (vstup/výstup)

- V datovém segmentu musí být připraven parametrický blok pro službu. Tento blok musí být zarovnaný na 4B (=align 2)

- v reg.R0L je uloženo číslo požadované služby, R0H=0x01
- v reg.ER1 je adresa parametrického bloku
- službu voláme skokem do podprogramu JSR @addr (addr je adresa nastvená v prostředí překladače jako „simulated i/o address“)

služba	č.služby (R0L)	Param.blok	Popis
GETC	0x11	Adresa vstupního bufferu (4B)	Čtení znaku z klávesnice
PUTC	0x12	Adresa výstupního bufferu (4B)	Výstup jednoho znaku na obrazovku
GETS	0x13	Adresa vstupního bufferu (4B)	Čtení/zadání řádku z klávesnice, posl. Uložený znak je 0x0a (=enter)
PUTS	0x14	Adresa výstupního bufferu (4B)	Výstup textu na obrazovku. Poslední znak je 0x0a nebo 0x00

Příklad užití

```
.h8300s
.equ syscall,0x1FF00          ; simulated IO address
.equ PUTS,0x0114             ; kod PUTS (0x01,0x14)
.equ GETS,0x0113             ; kod GETS (0x01,0x13)
; ----- datovy segment -----
.data
txt1:      .asciz "Zadejte vstup:\n" ; vypisovany text ukonceny \n (=0x0d 0x0a)
buffer:    .space 20                ; vstupni buffer, max.19znaku +enter

        .align 2                    ; zarovnani adresy (parametricke bloky musi byt zarovnane na 4B)
par1:     .long txt1                ; parametricky blok 1 - ukazatel na vystupni buffer
par2:     .long buffer              ; parametricky blok 2 - ukazatel na vstupni buffer

        .align 1                    ; zarovnani adresy (stack musi byt zarovnany na 2B (=word))
        .space 100                  ; stack
stck:     ; konec stacku + 1

; ----- kodovy segment -----

        .text
.global _start
_start:  mov.l #stck,ER7              ;stack je nutno definovat, pouzivame volani fci (JSR)
        mov.w #PUTS,R0               ; kod služby PUTS do R0H,R0L
        mov.l #par1,ER1              ; adr. param. bloku do ER1
        jsr @syscall                 ;volani syst.fce = vypsani textu txt1

        mov.w #GETS,R0               ; kod služby GETS do R0H,R0L
        mov.l #par2,ER1              ; adr. param. bloku do ER1
        jsr @syscall                 ; volani syst.fce = uzivatel zada text, který bude ulozen do buffer

        mov.l #vstup,ER6             ; adresa retezce do ER6 (vstupni param.fce)
        jsr @prevod                  ; volani funkce prevod

lab1:    jmp @lab1                    ; konec

; --- funkce prevod ASCII retezec->cislo
prevod:  ;vstup: ER6 = adresa retezce; vystup: R0 = cislo (0-65535)
        push.l ER1 ; ulozeni registru
        xor.w R0,R0
        xor.w R1,R1
        mov.w #10,E1
lab2:    mov.b @ER6,R1L ;precti znak z pameti
        cmp.b #0x0A,R1L ; test na konec (CR)
        beq lab3          ;je konec, tj. Znak = CR
        add.b #-'0',R1L    ;odečtení '0' (0x30)
        mulxu.w E1,ER0     ;ER0=10*ER0
        add.w R1,R0        ;R0=R0+R1
        inc.l #1,ER6       ;dalsi adresa (znak)
        jmp @lab2
lab3:    pop.l ER1 ; obnoveni registru (=puvodni hodnota)
        rts ; navrat z podprogramu

.end
```