Programowanie mikrokontrolerów
Inne architektury

Marcin Engel Marcin Peczarski

23 września 2010

Dlaczego MSP430?

- Bo mam taki hardware.
- Bo zainteresowała mnie ta architektura.
- Bo są dostępne darmowe narzędzia programistyczne.
- Aby porównać z architekturą AVR.

MSP430 – podstawowe informacje

- Rodzina mikrokontrolerów firmy Texas Instruments
- Przeznaczona do aplikacji o małym poborze prądu
- Architektura 16-bitowa
- Little-endian
- RISC – tylko 27 instrukcji
- Przestrzeń adresowa 64 kB
- Pamięć operacyjna od 128 B do 16 kB
- Pamięć nieulotna od 1 kB do 256 kB
- Wiele modeli w różnych obudowach – od 14 do 100 wyprowadzeń

MSP430 – układy peryferyjne

- Poszczególne modele są różnie wyposażone w wiele typowych dla mikrokontrolerów układów peryferyjnych.
- Do 12 8-bitowych portów wejścia-wyjścia
- Liczniki 8 i 16-bitowe
- Watchdog
- Komparator
- Wzmacniacz operacyjny
- Przetwornik analogowo-cyfrowy: 10/12-bitowy, 16-bitowy
- Przetwornik cyfrowo-analogowy 12-bitowy
- Interfejsy szeregowe: UART, I^2C, SPI, IrDA
- DMA
- Sterownik LCD
- Układ mnożący
- Czujnik temperatury

MSP430 – przykładowa przestrzeń adresowa

- `0x0000-0x00ff`
  Początkowe 256 B to 8-bitowe układy wejścia-wyjścia
- `0x0100-0x01ff`
  Kolejne 256 B to 16-bitowe układy wejścia-wyjścia
- `0x0200-0x09ff`
  2 kB pamięci RAM
- `0x0c00-0x0fff`
  1 kB pamięci ROM – fabrycznie zaprogramowany bootloader
- `0x1000-0x10ff`
  256 B pamięci FLASH dla danych
- `0x1100-0xffff`
  reszta z 60 kB pamięci FLASH dla programu

MSP430 – Asembler – rodzaje instrukcji

- Instrukcje dwuargumentowe:

  ```
  mnemonic source, destination
  ```

- Np. dodanie zawartości rejestru `r5` do zawartości rejestru `r7`:

  ```
  add r5, r7
  ```

- Instrukcje jednoargumentowe:

  ```
  mnemonic destination
  ```

- Np. odłożenie zawartości rejestru `r5` na stos:

  ```
  push r5
  ```

- Skoki warunkowe względne i bezwarunkowy względny, np.:

  ```
  jmp offset
  ```

MSP430 – Asembler – instrukcje 8-bitowe

- Instrukcje domyślnie wykonują się na argumentach 16-bitowych.
- Większość instrukcji arytmetycznych ma wersję 8-bitową, np.:

```
add.b r5, r7
```

- Instrukcje 8-bitowe zerują starsze 8 bitów argumentu docelowego.
- Poniższa instrukcja zeruje starszy bajt rejestru `r5`.

```
mov.b r5, r5
```

MSP430 – rejestry

- MSP430 posiada 16 rejestrów 16-bitowych.
- Rejestr `r0` to licznik programu!
- Zapis do `r0` powoduje wykonanie skoku!

```
br #0xa034; skrót dla mov #0xa034, r0
```

- Rejestr `r0` ma zawsze wartość parzystą.
- Rejestr `r1` jest wskaźnikiem stosu.
- Rejestr `r2` zawiera znaczniki.
- Rejestr `r3` służy do ładowania stałych.
- Odczyt z `r3` w zależności od trybu adresowania ładuje stałą, np. wyzerowanie rejestru `r5`:

```
mov #0, r5; skrót dla mov r3, r5
```

- Zapis do `r3` jest ignorowany.
- Rejestry `r4` do `r15` są rejestrami ogólnego przeznaczenia.

MSP430 – rejestr znaczników

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
zarezerwowane							V	SCG1	SCG0	OSCOFF	CPUOFF	GIE	N	Z	C

- Bity SCG1, SCG0, OSCOFF, CPUOFF służą do sterowania trybami oszczędzania energii.
- Bit GIE (ang. global interrupt enable) – służy do uaktywniania systemu przerwań.
- Bity V, N, Z, C to tradycyjne znaczniki, odpowiednio: nadmiar w arytmetyce uzupełnieniowej, wynik ujemny, wynik zerowy, przeniesienie-pożyczka.

MSP430 – tryby adresowania argumentu źródłowego

- Rejestrowy bezpośredni

```
mov r5, r7
```

  - Użycie rejestru `r3` oznacza ładowanie stałej 0.

```
mov #0, r7
```

- Rejestrowy indeksowy (pośredni)

```
mov 0xff(r5), r7
```

  - Użycie rejestru `r2` oznacza adresowanie bezpośrednie.

```
mov &0x12ab, r7
```

  - Użycie rejestru `r3` oznacza ładowanie stałej 1.

```
mov #1, r7
```

MSP430 – tryby adresowania argumentu źródłowego, cd.

- Rejestrowy pośredni

```
mov @r5, r7
```

  - Użycie rejestru `r2` oznacza ładowanie stałej 4.

```
mov #4, r7
```

  - Użycie rejestru `r3` oznacza ładowanie stałej 2.

```
mov #2, r7
```

- Rejestrowy pośredni z postinkrementacją

```
mov @r5+, r7
```

  - Użycie rejestru `r2` oznacza ładowanie stałej 8.

```
mov #8, r7
```

  - Użycie rejestru `r3` oznacza ładowanie stałej -1 (0xffff).

```
mov #-1, r7
```

MSP430 – tryby adresowania argumentu docelowego

- Rejestrowy bezpośredni

```
mov r5, r7
```

- Rejestrowy indeksowy (pośredni)

```
mov r5, 0xff(r7)
```

- Tryb rejestrowy pośredni jest realizowany przez podanie zerowego przesunięcia.

```
mov r5, 0(r7)
```

MSP430 – sztuczki z trybami adresowania

- Dla uzyskania trybu adresowania natychmiastowego stosuje się tryb rejestrowy pośredni z postinkrementacją z rejestrem `r0`.
- Instrukcja

```
mov #1234, r7
```

- jest tłumaczona na

```
mov @r0+, r7
.dw 1234
```

- Tak zaawansowane tryby adresowania umożliwiają bardzo efektywne kodowanie.
- Instrukcja z języka C

```
*p++ *= 2;
```

- może być przetłumaczona na jedną instrukcję Asemblera

```
add @r5+, -2(r5)
```

MSP430 – instrukcje dwuargumentowe

- Jest ich tylko 12:

```
mov  src, dst
add  src, dst
addc src, dst
subc src, dst
sub  src, dst
cmp  src, dst
dadd src, dst
bit  src, dst
bic  src, dst
bis  src, dst
xor  src, dst
and  src, dst
```

MSP430 – instrukcje jednoargumentowe

- Jest ich tylko 7:

```
rrc  dst ; rotacja w prawo o 1 bit
         ; przez bit przeniesienia
swpb dst ; zamiana miejscami bajtów
rra  dst ; arytmetyczne przesunięcie
         ; w prawo o 1 bit
sxt  dst ; rozszerzenie ze znakiem
         ; wartości 8- do 16-bitowej
push dst
call dst
reti     ; powrót z przerwania
         ; argument nieużywany
```

MSP430 – instrukcje skoków

- Jest ich tylko 8:

```
jne/jnz offset ; skok gdy Z == 0
               ; wartości równe
jeq/jz  offset ; skok gdy Z == 1
               ; wartości różne
jnc/jlo offset ; skok gdy C == 0
               ; < bez znaku
jc/jhs  offset ; skok gdy C == 1
               ; >= bez znaku
jn      offset ; skok gdy N == 1
               ; wynik ujemny
jge     offset ; skok gdy N == V
               ; >= ze znakiem
jl      offset ; skok gdy N != V
               ; < ze znakiem
jmp     offset ; skok bezwarunkowy
```

MSP430 – pseudoinstrukcje

- Razem mamy więc tylko 27 instrukcji.
- Pozostałe potrzebne instrukcje realizowane są przez wykorzystanie trybów adresowania, np.:

```
nop                 mov  r3, r3
pop  dst            mov  @r1+, dst
br   dst            mov  dst, r0
ret                 mov  @r1+, r0
rla  dst            add  dst, dst
rlc  dst            addc dst, dst
inv  dst            xor  #-1, dst
clr  dst            mov  #0, dst
tst  dst            cmp  #0, dst
dec  dst            sub  #1, dst
decd dst            sub  #2, dst
inc  dst            add  #1, dst
incd dst            add  #2, dst
```

MSP430 – pseudoinstrukcje, cd.

- Bitami w rejestrze znaczników można manipulować za pomocą następujących instrukcji:

`clrc`	`bic #1, r2`
`setc`	`bis #1, r2`
`clrz`	`bic #2, r2`
`setz`	`bis #2, r2`
`clrn`	`bic #4, r2`
`setn`	`bis #4, r2`
`dint`	`bic #8, r2`
`eint`	`bis #8, r2`

MSP430 – cykl rozkazowy

- Każdy dostęp do pamięci wymaga jednego cyklu zegara.
- Jeden cykl zajmuje pobranie instrukcji.
- Dodatkowy cykl potrzebny jest, jeśli argument źrodłowy jest w pamięci.
- Dodatkowe dwa cykle są potrzebne, jeśli argument docelowy jest w pamięci.
- Jeszcze jeden dodatkowy cykl jest potrzebny na każdy argument z adresowaniem indeksowym.
- Instrukcja skoku względnego wykonuje się dwa cykle zegara niezależnie od spełnienia warunku.
- Reasumując, cykl wykonywania instrukcji może trwać od jednego do sześciu cykli zegara.

MSP430 – przerwania

- Jest 16 źródeł przerwań.
- 14 źródeł jest maskowanych bitem GIE w rejestrze znaczników, a 2 źródła są niemaskowane (NMI, RST).
- Wektor przerwań jest umieszczony w końcowych 32 bajtach przestrzeni adresowej.
- Element wektora przerwań przechowuje adres procedury jego obsługi.
- Wektor resetu (RST) znajduje się pod adresem 0xfffe.
- Przerwanie o większym adresie wektora przerwań ma wyższy priorytet.
- Zgłoszenie przerwania trwa 6 cykli zegara.
- Z wyjątkiem resetu po zgłoszeniu przerwania na stos odkładany jest adres powrotu i rejestr znaczników.
- Przejście do procedury obsługi przerwania blokuje przerwania (bit GIE) i tryby oszczędzania energii (bit SCG0).

MSP430 – tryby oszczędzania energii

- Jest to architektura specjalnie zaprojektowana z myślą o zasilaniu bateryjnym.
- Zakres napięć zasilania wynosi 1,8–3,6 V.
- Ma prosty układ sterujący: mało instrukcji, brak potokowości, w pełni deterministyczny cykl rozkazowy.
- Jest pięć trybów oszczędzania energii.
- Do mikrokontrolera podłącza się zwykle dwa kwarce:
  - zegarkowy 32768 kHz,
  - o dużej częstotliwości, np. 6 MHz.

MSP430 – prosty program

```
#include <io.h>

void wait(void) {
  volatile int i;
  for (i = 0; i < 32000; ++i);
}
int main() {
  P4DIR = 0xE;   /* rejestr kierunku portu P4 */
  for (;;) {
    P4OUT = 0xC; /* włączenie diody zielonej  */
    wait();
    P4OUT = 0xA; /* włączenie diody żółtej    */
    wait();
    P4OUT = 0x6; /* włączenie diody czerwonej */
    wait();
  }
}
```