ESDA LAB

Ανάλυση Παραδείγματος ARM9 Assembly Μέρος 1ο

Αλέξανδρος Σπουρνιάς
PhD Candidate

Εργαστήριο Σχεδιασμού Ενσωματωμένων Συστημάτων & Εφαρμογών

Γενική Επισκόπηση

Αυτό το εργαστήριο αφορά τον **προγραμματισμό GPIOs (General Purpose Input/Output)** στο μικροελεγκτή **STR91x** της STMicroelectronics, χρησιμοποιώντας **γλώσσα Assembly**. Οι στόχοι περιλαμβάνουν:

1. **Κατανόηση της αρχιτεκτονικής ARM (STR91x):**
   Διαχείριση των καταχωρητών (registers) και του υλικού (hardware) σε χαμηλό επίπεδο.
2. **Ρύθμιση και χρήση GPIO θυρών:**
   1. Διαμόρφωση εισόδων/εξόδων.
   2. Ενεργοποίηση ρολογιών (clocks).
   3. Ανάγνωση από pin εισόδου.
   4. Εγγραφή σε pin εξόδου.
3. **Γραφή Assembly κώδικα για ελεγχόμενες λειτουργίες I/O:**
   1. Άνοιγμα LED με κουμπί.
   2. Παρουσίαση καθυστέρησης (delays) μέσω βρόχων.
   3. Εκτέλεση χωρίς λειτουργικό σύστημα ή βιβλιοθήκες υψηλού επιπέδου.

Lab6_GPIOs_assembly.s

```
APB0_NBUF_BASE  EQU  0x58000000    ; APB Bridge 0 Non-buffered
APB1_NBUF_BASE  EQU  0x5C000000    ; APB Bridge 1 Non-buffered
```

Ο **APB,** είναι ο δίαυλος μέσω του οποίου η CPU επικοινωνεί με περιφερειακά όπως GPIO, Timers, UARTs. Το STR91x έχει δύο γεφυρώσεις (bridges):

1. APB0: για GPIO και άλλα βασικά περιφερειακά.
2. APB1: για ρυθμίσεις συστήματος (System Control Unit).

Η εντολή **EQU (equate)** χρησιμοποιείται για να δηλώσουμε **σταθερές τιμές (constants)**, οι οποίες δεν αλλάζουν κατά την εκτέλεση του προγράμματος. Όταν γράφουμε: APB0_NBUF_BASE EQU 0x58000000, σημαίνει ότι κάθε φορά που ο assembler δει APB0_NBUF_BASE, να το αντικαθιστά με τη διεύθυνση 0x58000000».

Lab6_GPIOs_assembly.s

Το **SCU (System Control Unit)** είναι υποσύστημα υπεύθυνο για τον **έλεγχο ρολογιών** (clocks) στα περιφερειακά, την **απελευθέρωση από reset,** την **παροχή ειδικών ρυθμίσεων** για GPIO, Flash, κ.λπ.

Προσφέρει πρόσβαση σε ειδικά καταχωρημένους registers, όπου ελέγχουν αν κάθε περιφερειακό έχει ενεργοποιηθεί σωστά.

```
SCU_BASE  EQU  APB1_NBUF_BASE + 0x2000
```

Υπολογίζει τη **βάση διεύθυνσης (base address)** του SCU:

1. APB1_NBUF_BASE = 0x5C000000
2. Προσθέτει 0x2000 → 0x5C002000

Άρα, το SCU ξεκινά από **διεύθυνση 0x5C002000** στην μνήμη, μέσω του διαύλου APB1.

Lab6_GPIOs_assembly.s

SCU_PCGR1_OFS EQU 0x18

Το SCU_PCGR1_OFS είναι ένας **offset**, δηλαδή η απόσταση (σε bytes) από τη βάση της διεύθυνσης του SCU, για να φτάσουμε στο **Peripheral Clock Gating Register 1**.

SCU_BASE = 0x5C002000

PCGR1 offset = 0x18

→ 0x5C002000 + 0x18 = 0x5C002018

Στον STR91x, κάθε περιφερειακό (π.χ. GPIO, UART, Timer, κ.λπ.) έχει τη δική του πηγή ρολογιού (clock source), όπου Για λόγους εξοικονόμησης ενέργειας, αυτά τα ρολόγια είναι απενεργοποιημένα εξ αρχής.

Εάν δεν ενεργοποιηθεί το ρολόι για ένα περιφερειακό, τότε το περιφερειακό ναι μεν υπάρχει στη μνήμη, αλλά η CPU δεν μπορεί να το διαβάσει, το γράψει, να το ελέγξει.

Lab6_GPIOs_assembly.s

SCU_PRR1_OFS EQU 0x20

Αυτό δηλώνει ότι ο **Peripheral Reset Register 1 (PRR1)** βρίσκεται **20 bytes μετά** από τη βάση του SCU. Ο **PRR1** είναι ένας register που ελέγχει το σήμα **Reset** για πολλά περιφερειακά.
Με bit **0** σε αυτό το register, το περιφερειακό παραμένει σε reset κατάσταση (σαν να είναι απενεργοποιημένο), ενώ με bit **1,** το περιφερειακό βγαίνει από reset και μπορεί να λειτουργήσει.

SCU_BASE = 0x5C002000

Offset = 0x20

→ Διεύθυνση PRR1 = 0x5C002020

Lab6_GPIOs_assembly.s

SCU_GPIOUT7_OFS EQU 0x60

Ορίζει ότι το **offset** για τον καταχωρητή εξόδου **GPIO7** (Output Register) είναι **0x60** bytes μετά τη βάση του SCU. Αυτός ο καταχωρητής, χρησιμοποιείται για να δηλώσουμε ποια από τα pins του GPIO7 λειτουργούν ως **ψηφιακές έξοδοι** (και όχι είσοδοι ή εναλλακτικές λειτουργίες).

Ο **Port 7** είναι ένα σύνολο από **ψηφιακά pins,** (π.χ. P7.0 έως P7.7).
Κάθε bit στον register αντιστοιχεί σε ένα pin:
bit 0 → P7.0
bit 1 → P7.1
…
bit 7 → P7.7

SCU_BASE = 0x5C002000

Offset = 0x60

→ 0x5C002000 + 0x60 = 0x5C002060

Lab6_GPIOs_assembly.s

```
SCU_PCGR1_Val   EQU  0x00220000
SCU_PRR1_Val    EQU  0x00220000
```

Δηλώνει τη **μάσκα** για να **αφαιρεθεί το reset** από τα ίδια περιφερειακά μέσω του PRR1 register.
Στην ουσία, είναι το "ξεκλείδωμα" που επιτρέπει στο περιφερειακό να λειτουργήσει μετά την ενεργοποίηση του ρολογιού.
Και στις δύο περιπτώσεις, χρησιμοποιείται η ίδια τιμή, γιατί αφορά **τα ίδια bits** (τα ίδια περιφερειακά).

1. PCGR1 (Clock): Ξεκλειδώνει ρολόι για GPIO.
2. PRR1 (Reset): Βγάζει GPIO από reset.

Lab6_GPIOs_assembly.s

GPIO7_DATA_ADDR EQU APB0_NBUF_BASE + 0x0000D000

Αυτή η γραμμή δηλώνει τη βασική διεύθυνση μνήμης του καταχωρητή δεδομένων (data register) για τον Port 7 (GPIO7).
Το **GPIO7_DATA_ADDR,** είναι η διεύθυνση από όπου η CPU διαβάζει την κατάσταση **(0** ή **1)** των pins ή γράφει τιμές για να θέσει κάποιο pin ψηλά/χαμηλά (High/Low).

GPIO7_DIR_OFS EQU 0x00000400

Δηλώνει το **offset** για τον καταχωρητή κατεύθυνσης (direction register) του Port 7.
Αυτός ο register, καθορίζει τι είδους λειτουργία έχει το κάθε pin, δηλαδή, έξοδος (output), ή είσοδος (input).

Μαζί, αυτοί οι δύο register επιτρέπουν στον μικροελεγκτή, να επικοινωνεί με το εξωτερικό περιβάλλον, όπως π.χ. τα LED, τα κουμπιά, οι αισθητήρες.

Lab6_GPIOs_assembly.s

SCU_GPIOUT7_Val EQU 0x00005555

Δηλώνει ότι η σταθερά SCU_GPIOUT7_Val έχει την τιμή 0x00005555.
Αυτή η τιμή θα φορτωθεί στον GPIO Output Mode Register του **Port 7**, για να ρυθμίσει τη λειτουργία κάθε pin ως **"Normal Output (mode 1)»**

Hex: 0x00005555

Bin: 0000 0000 0000 0000 0101 0101 0101 0101

↑ ↑ ↑ ↑ ↑ ↑ ↑ ↑

Bit: P7.7 ..... P7.0

Κάθε **2 bits** αντιστοιχούν **σε 1 pin του GPIO7** και καθορίζουν τη **λειτουργία (mode)** του.

Η γραμμή SCU_GPIOUT7_Val EQU 0x00005555 εξασφαλίζει ότι **όλα τα pins του GPIO7** θα λειτουργούν σωστά ως έξοδοι, με σταθερή λογική push-pull που επιτρέπει αξιόπιστο ψηφιακό έλεγχο.

Lab6_GPIOs_assembly.s

GPIO7_DIR_Val EQU 0xFF

Ορίζει τη σταθερά GPIO7_DIR_Val με την τιμή 0xFF, που σημαίνει ότι Κάθε bit του Port 7 ρυθμίζεται ως **έξοδος (output)**.

Hex: 0xFF

Bin: 11111111

Συμπερασματικά,

1. Το SCU λέει **τι είδους έξοδος** είναι (λογική push-pull)
2. Ο DIR λέει **αν είναι έξοδος ή είσοδος**

Χρειάζονται και τα δύο βήματα για να λειτουργήσει σωστά το GPIO.

Lab6_GPIOs_assembly.s

GPIO3_DATA_ADDR EQU APB0_NBUF_BASE + 0x00009000

Ορίζει μια σταθερή διεύθυνση μνήμης για τον καταχωρητή δεδομένων (data register) του **Port 3 (GPIO3)** και αντιπροσωπεύει τη θέση στη μνήμη από όπου η CPU μπορεί να **διαβάσει ή να γράψει bits** από και προς, τα pins του GPIO3.

Ο καταχωρητής GPIO3_DATA_ADDR χρησιμοποιείται για την **ανάγνωση** της κατάστασης εισόδων (π.χ. κουμπιών), καθώς και την **εγγραφή** της τιμής, αν τα pins είναι έξοδοι.

Συνολικά, η γραμμή GPIO3_DATA_ADDR EQU 0x58009000 ορίζει τη **διεύθυνση πρόσβασης στο Port 3**, μέσω της οποίας η CPU μπορεί να **διαβάσει την κατάσταση ενός κουμπιού** ή οποιουδήποτε εξωτερικού σήματος συνδεδεμένου στον GPIO3.

Lab6_GPIOs_assembly.s

AREA Reset, CODE, READONLY

Το **AREA,** δημιουργεί μία **λογική περιοχή κώδικα** με το όνομα Reset, το **CODE,** ορίζει ότι η περιοχή περιέχει εντολές (instruction section), ενώ το **READONLY,** ορίζει ότι δεν θα αλλάζει (read-only memory).

Τοποθετείται **στη διεύθυνση 0x00000000** όπου είναι το σημείο από το οποίο ξεκινά η CPU μετά από **reset.**

ENTRY

Το **ENTRY,** δηλώνει το **σημείο εκκίνησης του προγράμματος** (entry point), ώστε ο linker να ξέρει από πού να ξεκινήσει την εκτέλεση. Στην ουσία, αυτός είναι ο reset vector του προγράμματος.

Lab6_GPIOs_assembly.s

```
NOP     ; Wait for OSC stabilization
        NOP
        NOP
        .....
        NOP
```

Το **NOP,** (**No Operation),** δεν κάνει τίποτα απολύτως, εκτός από το να καταναλώνει λίγο χρόνο CPU. **Χρησιμοποιείται** αμέσως μετά το reset, καθώς το **ρολόι (oscillator)** χρειάζεται μερικά cycles για να σταθεροποιηθεί.
Χωρίς αυτό το "buffer" καθυστέρησης, μπορεί ο μικροελεγκτής να συνεχίσει την εκτέλεση, **πριν το ρολόι να είναι έτοιμο**, οδηγώντας σε απροσδόκητη συμπεριφορά.
10 NOPs = ασφαλές χρονικό διάστημα (μερικά μικροδευτερόλεπτα)

Lab6_GPIOs_assembly.s

LDR R0, =SCU_BASE

Η εντολή αυτή, φορτώνει στον καταχωρητή **R0** τη σταθερά της διεύθυνσης **SCU_BASE**

SCU_BASE = 0x5C002000
→ R0 = 0x5C002000

Η εντολή LDR R0, =SCU_BASE φορτώνει τη βάση διεύθυνσης του System Control Unit στον καταχωρητή R0. Αυτό επιτρέπει να γίνονται προσπελάσεις σε πολλούς SCU registers με **απλά offsets**, κάνοντας τον κώδικα πιο δομημένο και ευέλικτο.

Lab6_GPIOs_assembly.s

LDR R1, =SCU_PCGR1_Val

Φορτώνει στον R1 τη σταθερή τιμή SCU_PCGR1_Val = 0x00220000
Στην ουσία, αυτή είναι η μάσκα ενεργοποίησης του ρολογιού για τα περιφερειακά (π.χ. GPIO3 & GPIO7), και καθορίζει ποια bits του Peripheral Clock Gating Register 1 (PCGR1) θα γίνουν 1.

STR R1, [R0, #SCU_PCGR1_OFS]

Γράφει την τιμή του R1 (0x00220000) στη διεύθυνση:
R0 = SCU_BASE = 0x5C002000
SCU_PCGR1_OFS = 0x18
Τελική διεύθυνση: 0x5C002018
Έτσι, ενεργοποιούμε το **ρολόι (clock enable)** για τα αντίστοιχα περιφερειακά, γράφοντας στο PCGR1.

Εν κατακλείδι, αυτές οι δύο γραμμές, είναι το σημείο όπου το πρόγραμμα **ενεργοποιεί επίσημα τα περιφερειακά** GPIO3 και GPIO7, επιτρέποντάς τους να λειτουργήσουν.

Lab6_GPIOs_assembly.s

LDR R1, =SCU_PRR1_Val

Φορτώνει στον R1 τη σταθερή τιμή SCU_PPR1_Val = 0x00220000

Πρόκειται για μάσκα bits που αντιστοιχεί στα περιφερειακά GPIO3 και GPIO7 και καθορίζει ποια modules θα **βγουν από reset (bit = 1)**.

STR R1, [R0, #SCU_PRR1_OFS]

Γράφει τη μάσκα 0x00220000 στη διεύθυνση:
R0 = SCU_BASE = 0x5C002000
SCU_PRR1_OFS = 0x20
→ 0x5C002000 + 0x20 = 0x5C002020
Έτσι, ενημερώνουμε το **Peripheral Reset Register 1 (PRR1),** ώστε να **αφαιρέσουμε το reset** από τα GPIO3 και GPIO7.
Εν κατακλείδι, αυτές οι δύο γραμμές, είναι το σημείο όπου το πρόγραμμα αφαιρεί το reset από τα περιφερειακά GPIO3 και GPIO7, επιτρέποντάς τους να ενεργοποιηθούν πλήρως.

Lab6_GPIOs_assembly.s

LDR R1, =SCU_GPIOUT7_Val

Φορτώνει στον R1 την τιμή 0x00005555, όπου κάθε pin του Port 7, ρυθμίζεται σε **Mode 1 (Normal Output / Push-Pull)**.
(0101010101010101, όλα τα pins έχουν mode bits 01).

STR R1, [R0, #SCU_GPIOUT7_OFS]

Γράφει την τιμή 0x00005555 στη διεύθυνση:

R0 = SCU_BASE = 0x5C002000

SCU_GPIOUT7_OFS = 0x60

Διεύθυνση: 0x5C002060

Όπου αυτός είναι ο καταχωρητής **SCU Output Mode για τον GPIO7**.

Ενεργοποιούμε έτσι **τη δυνατότητα των pins του GPIO7 να βγάζουν σήμα (3.3V ή 0V)** προς το εξωτερικό κύκλωμα (π.χ. LED).

Lab6_GPIOs_assembly.s

```
LDR     R0, =GPIO7_DATA_ADDR
LDR     R1, =GPIO7_DIR_Val
STR     R1, [R0, #GPIO7_DIR_OFS]
```

Φορτώνει στον R0 τη διεύθυνση του καταχωρητή δεδομένων του GPIO7.
Φορτώνει στον R1 την τιμή 0xFF (δηλαδή όλα τα pins του Port 7 είναι έξοδοι).
Γράφει αυτήν την τιμή στον **direction register του GPIO7**, ο οποίος καθορίζει **αν κάθε pin είναι είσοδος ή έξοδος**.

Αυτό το block κώδικα, καθορίζει με ακρίβεια ότι **όλα τα pins του GPIO7 θα λειτουργούν ως έξοδοι**, επιτρέποντας στον μικροελεγκτή να αποστέλλει ψηφιακά σήματα (0/1) προς το εξωτερικό κύκλωμα, (πχ για να ανάβει τα LEDs)

Lab6_GPIOs_assembly.s

```
Check_Button
   LDR    R0, =GPIO3_DATA_ADDR     ; Διεύθυνση GPIO3
   LDRB   R0, [R0, #0x080]         ; Φόρτωσε 1 byte με offset (bitmask με P3.5)
   CMP    R0, #0x00                ; Είναι πατημένο (LOW);
   BNE    Led0_OFF                 ; Αν είναι ≠0 → δεν είναι πατημένο → σβήσε LED
   B      Led0_ON                  ; Αλλιώς → άναψε LED
```

1. LDR R0, =GPIO3_DATA_ADDR → φορτώνει στον R0 τη διεύθυνση του GPIO3
2. LDRB = **(Load Register Byte)** → διαβάζει 8 bits (1 byte), #0x080 είναι η μάσκα που περιέχει το **bit 5** (GPIO3.5)
3. CMP → συγκρίνει αν είναι 0 (πατημένο)
4. BNE - (Branch if Not Equal) → **αν δεν είναι 0, πήγαινε στο Led0_OFF**
5. B Led0_ON - (branch) → Αλλιώς άναψε LED

Lab6_GPIOs_assembly.s

```
Led0_ON
   LDR    R0, =GPIO7_DATA_ADDR
   LDR    R1, =0x01                    ; Bit 1 HIGH → άναψε LED0
   STR    R1, [R0, #0x004]
   B      Check_Button                 ; Επανάλαβε
```

1. LDR R0, =GPIO7_DATA_ADDR → φορτώνει στο R0 τη βασική διεύθυνση του GPIO7
2. LDR R1, =0x01 → φορτώνει στο R1 την τιμή 0x01 (δηλαδή Bit 0 HIGH = άναψε LED0)
3. STR R1, [R0, #0x004] → γράφει την τιμή στο GPIO7 output register (0x5800D004) και ανάβει LED
4. B Check_Button → επιστρέφει στον βρόχο για να ξαναελέγξει την κατάσταση του κουμπιού

Lab6_GPIOs_assembly.s

```
Led0_OFF
   LDR    R0, =GPIO7_DATA_ADDR
   LDR    R1, =0x00                    ; Bit 0 LOW → σβήσε LED0
   STR    R1, [R0, #0x004]
   B      Check_Button                 ; Επανάλαβε
```

1. LDR R0, =GPIO7_DATA_ADDR → φορτώνει στο R0 τη βασική διεύθυνση του GPIO7
2. LDR R1, =0x00 → φορτώνει στο R1 την τιμή 0x00 (δηλαδή Bit 0 LOW = σβήσε LED0)
3. STR R1, [R0, #0x004] → γράφει 0 στο GPIO7 output register (0x5800D004) και σβήνει το LED
4. B Check_Button → επιστρέφει στον έλεγχο κουμπιού για επανάληψη

ESDA LAB

Ερωτήσεις

Εργαστήριο Σχεδιασμού Ενσωματωμένων Συστημάτων & Εφαρμογών