

# ΕΡΓΑΣΤΗΡΙΟ ΒΑΣΙΚΑ ΘΕΜΑΤΑ ΑΡΧΙΤΕΚΤΟΝΙΚΗΣ ΥΠΟΛΟΓΙΣΤΩΝ


## ΣΥΜΒΟΛΙΚΗ ΓΛΩΣΣΑ - ΣΗΜΕΙΩΣΕΙΣ

Γ' Εξάμηνο, (Έκδοση 01/2018)

Νικόλαος Σκλάβος

e-mail : nsklavos AT ceid DOT upatras DOT gr

1



### Διάρθρωση Παρουσίασης

1. ΑΤ91: Περιγραφή, Λειτουργία, Χαρακτηριστικά
2. Εντολές
3. Αριθμητικές-Λογικές Πράξεις
4. ΑΤ91: Assembly
5. Εξομοιωτής Keil
6. Βιβλιογραφία

2

# AT91: ΠΕΡΙΓΡΑΦΗ, ΧΑΡΑΚΤΗΡΙΣΤΙΚΑ, ΛΕΙΤΟΥΡΓΙΑ

Ενότητα 1

3

## Πλατφόρμα AT91

- Εμπορικά Διαθέσιμη Αναπτυξιακή Πλακέτα:
  - AT91SAM9261EK της εταιρείας ATMEL.
- Υποστηρίζει την ανάπτυξη εφαρμογών υλικού και λογισμικού για το μικροελεγκτή AT91SAM9261:
  - εμπλουτισμένη υλοποίηση του ARM926EJ-S της εταιρείας ARM (Advanced Risc Machines)
- Ο μικροελεγκτής χρονίζεται με ένα ρολόι 200 MHz.

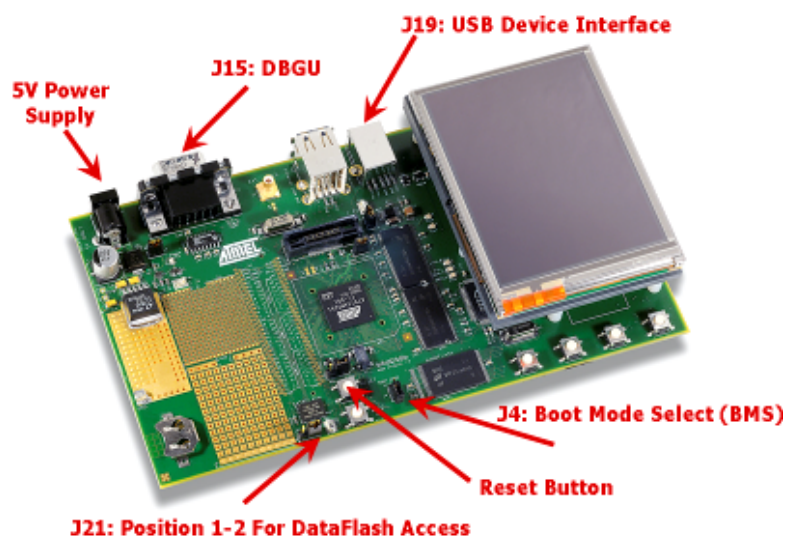
4

## Ιδιαίτερα χαρακτηριστικά του ARM9EJ-S

- **Reduced Instruction Set Computer (RISC)**  
επεξεργαστής: εξαιρετικά λίγες και απλές εντολές.
- Προσφέρει δύο διαφορετικά σύνολα εντολών:  
**ARM / THUMB.**
- **Ειδικές εντολές:**
  1. για γρήγορη επεξεργασία ψηφιακού σήματος (DSP)
  2. για την υλοποίηση της Jazelle αρχιτεκτονικής για JAVA

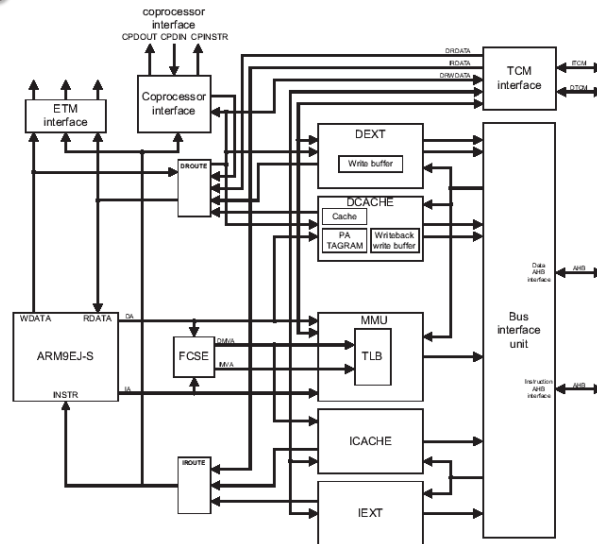
5

## Εικόνα Αναπτυξιακού



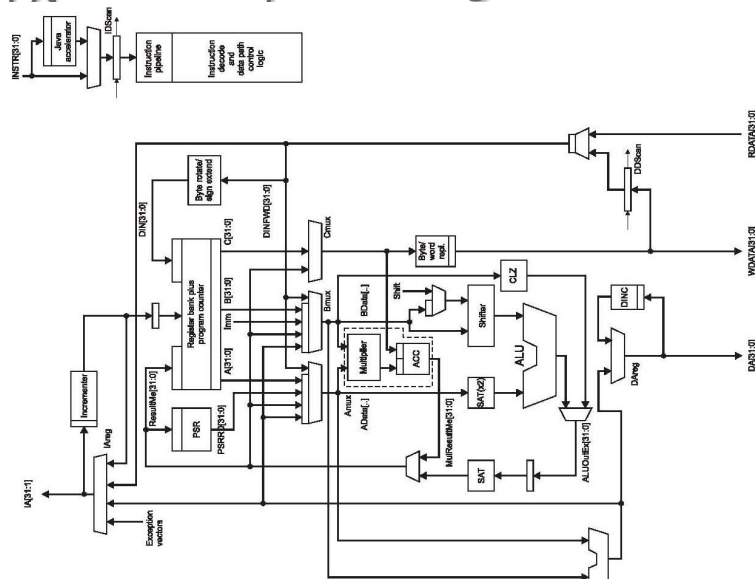
6

## Αρχιτεκτονική του επεξεργαστή ARM926EJ-S



7

## Αρχιτεκτονική του ARM9EJ-S



8

## Στοιχεία Επεξεργαστή:

- **Καταχωρητές (Registers):**
  - μικρές μνήμες εύρους 32-bit
- **Αριθμητική /Λογική Μονάδα (Arithmetic Logic Unit)**
- **Καταστάσεις Λειτουργίας:** επτά διαφορετικές καταστάσεις, όπου η διαφοροποίηση τους σχετίζεται με τα δικαιώματα πρόσβασης στις περιοχές της *εξωτερικής μνήμης*.

9

## Processor Modes

- **User** – κατάσταση εκτέλεσης προγραμμάτων
- **FIQ** – κατάσταση μετά από Fast Interrupt
- **IRQ** – κατάσταση μετά από Interrupt
- **Supervisor** – κατάσταση με αυξημένα δικαιώματα
- **Abort** – κατάσταση μετά από λάθος προσπέλαση στη μνήμη
- **Undefined** – κατάσταση μετά από εκτέλεση εντολής που δεν ανήκει στο σύνολο εντολών
- **System** – κατάσταση με αυξημένα δικαιώματα

10

## Καταχωρητές

- Ο επεξεργαστής διαθέτει 37 καταχωρητές:
  - 31 καταχωρητές γενικού σκοπού
  - 6 καταχωρητές ειδικού σκοπού
- Μόνο οι 17 είναι διαθέσιμοι και προσπελάσιμοι ανά πάσα στιγμή.
- Οι πρώτοι 8 (r0 έως r7) είναι κοινοί για όλες τις καταστάσεις του επεξεργαστή, ενώ οι υπόλοιποι 7 γίνονται διαθέσιμοι ανάλογα με την κατάσταση στην οποία βρίσκεται ο επεξεργαστής.

11

## Διαθέσιμοι Καταχωρητές – Κατάσταση Λειτουργίας

System and User	FIQ	Supervisor	Abort	IRQ	Undefined
r0	r0	r0	r0	r0	r0
r1	r1	r1	r1	r1	r1
r2	r2	r2	r2	r2	r2
r3	r3	r3	r3	r3	r3
r4	r4	r4	r4	r4	r4
r5	r5	r5	r5	r5	r5
r6	r6	r6	r6	r6	r6
r7	r7	r7	r7	r7	r7
r8	r8_fiq	r8	r8	r8	r8
r9	r9_fiq	r9	r9	r9	r9
r10	r10_fiq	r10	r10	r10	r10
r11	r11_fiq	r11	r11	r11	r11
r12	r12_fiq	r12	r12	r12	r12
r13	r13_fiq	r13_svc	r13_abt	r13_irq	r13_und
r14	r14_fiq	r14_svc	r14_abt	r14_irq	r14_und
r15 (PC)	r15 (PC)	r15 (PC)	r15 (PC)	r15 (PC)	r15 (PC)

Καταχωρητές κατάστασης του ARM

CPSR	CPSR	CPSR	CPSR	CPSR	CPSR
	SPSR_fiq	SPSR_svc	SPSR_abt	SPSR_irq	SPSR_und

12

## Χαρακτηριστικά Καταχωρητών

- **Εύρος 32-bit**, ισοδυναμεί με μια λέξη (word):
  - Γίνεται χρήση των 16 λιγότερο σημαντικών ψηφίων (half word) ή των 8 λιγότερο σημαντικών ψηφίων (byte).
- **User Mode:** χρησιμοποιούνται οι πρώτοι 13 καταχωρητές **r0 – r12**, για γενικό σκοπό
- **Οι επόμενοι τρεις έχουν συγκεκριμένη λειτουργία:**
  - **r13** δείκτης σωρού (stack pointer)
  - **r14** καταχωρητής διασύνδεσης κώδικα (branch και link)
  - **r15** μετρητής προγράμματος (program counter - pc)

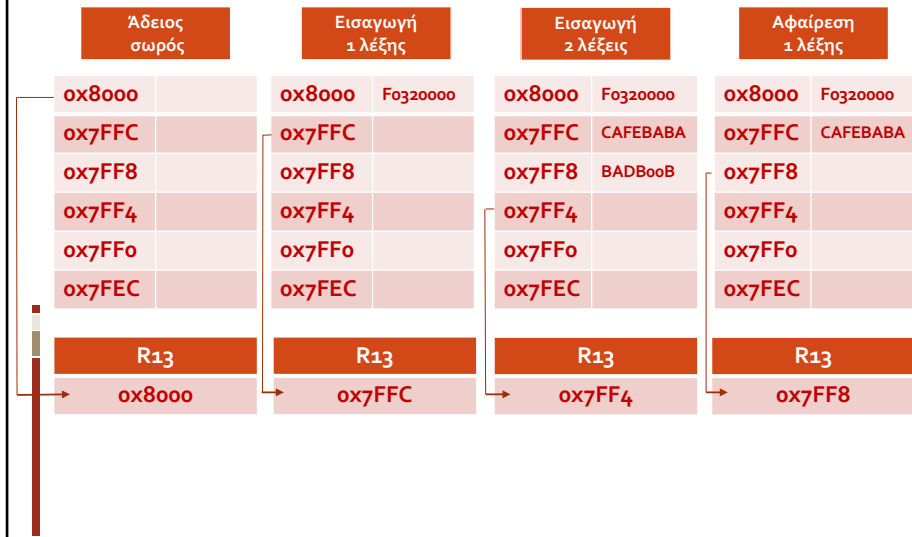
13

## Δείκτης Σωρού (Stack Pointer) – R13

- Χρησιμοποιείται για την διαχείριση του σωρού και έχει σαν περιεχόμενο τη διεύθυνση της μνήμης με την πρώτη ελεύθερη θέση του σωρού.
- Κάθε θέση στο σωρό έχει μέγεθος 32-bit (4-byte)
- **LIFO (Last In First Out):** ξεκινά από μια διεύθυνση και όσο γεμίζει με δεδομένα η θέση αυτή μειώνεται.
- Μειώνεται κατά 4 κατά την πρόσθεση μιας λέξης 32-bit
- Αυξάνεται κατά 4 κατά την απομάκρυνση μιας λέξης 32-bit

14

## Παράδειγμα Σωρού



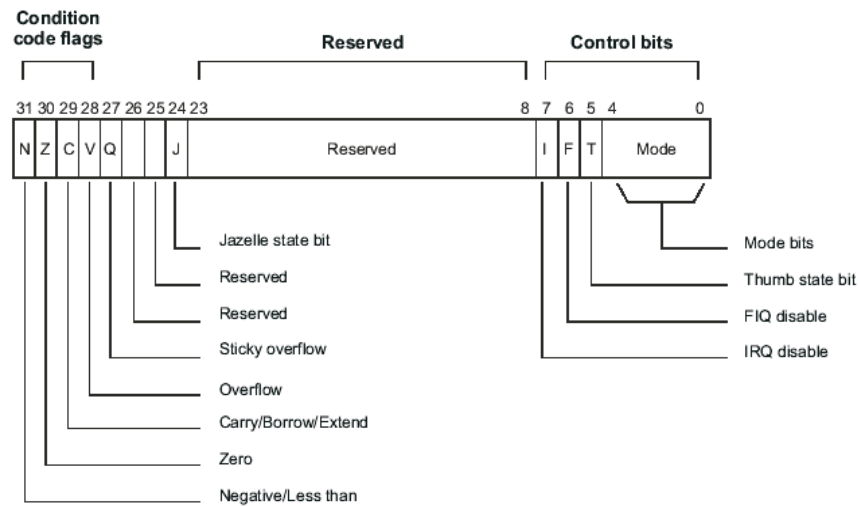
## Μετρητής Προγράμματος (Program Counter) – R14

- Περιέχει τη διεύθυνση στην οποία υπάρχει η επόμενη προς εκτέλεση εντολή.
- Κάθε εντολή έχει μέγεθος 32-bit (4-byte).
- Αυξάνεται κατά 4 κατά την εκτέλεση μιας εντολής 32-bit.
- Το περιεχόμενο του PC μπορούμε να το μεταβάλουμε και εμείς γράφοντας μια τιμή σε αυτόν, αρκεί να είμαστε σίγουροι ότι η τιμή που γράψαμε αντιστοιχεί σε μια διεύθυνση στην οποία υπάρχει κώδικας προς εκτέλεση.

16



## Καταχωρητής Κατάστασης: Current Processor Status Register (CPSR)



# ΕΝΤΟΛΕΣ

## Ενότητα 2

## Κατηγορίες Εντολών

1. **Μεταφορά δεδομένων**: από και προς την κύρια μνήμη
2. **Αριθμητικές πράξεις**: πρόσθεση, πολλαπλασιασμός, ...
3. **Λογικές πράξεις**: AND, OR, XOR, ...
4. **Διακλαδώσεις**: αλλαγή ροής εκτέλεσης κώδικα
5. **Συστήματος**: προσπέλαση συνεπεξεργαστή, κ.α.

19

## Εκτέλεση εντολής υπό συνθήκη

Συνθήκη	Περιγραφή	Συνθήκη	Περιγραφή
CS/HS	C=1 (Unsigned Higher or Same)	GE	N=V(Signed Greater Than or Equal)
CC/LO	C=0 (Unsigned Lower)	GT	Z=0 και N=V (Signed Greater Than)
EQ	Z=1 (Equal)	HI	C=1 και Z=0 (Unsigned Higher)
NE	Z=0 (Not Equal)	LE	Z=1 ή N!=V (Signed Less Than or Equal)
VS	V=1	LT	N!=V (Signed Less Than)
VC	V=0	LS	C=0 ή Z=1 (Unsigned Lower or Same)
MI	N=1 (Minus)		
PL	N=0 (Plus)		

Παράδειγμα:

Η εντολή **ADDCS R0, R1, R2** θα εκτελεστεί η πρόσθεση μόνο αν το Carry CPSR bit είναι ίσο με 1.

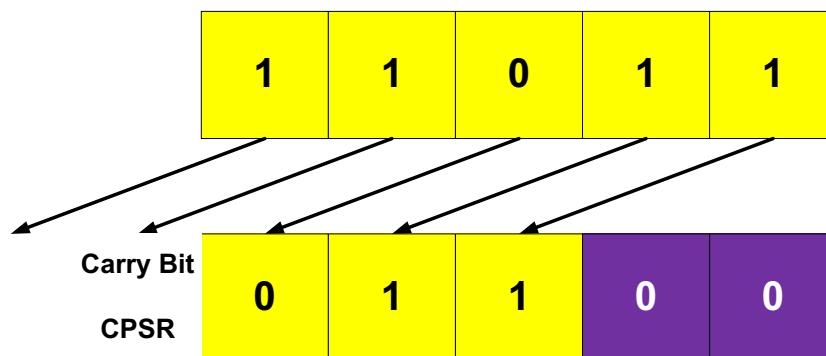
20

## Εντολές Ολίσθησης

- **Logic Shift Left (LSL)**: λογική ολίσθηση προς τα αριστερά
  - ισοδυναμεί με **πολλαπλασιασμό επι 2**
- **Logic Shift Right (LSR)**: λογική ολίσθηση προς τα δεξιά
  - ισοδυναμεί με **ακέραια διαίρεση δια 2**
- **Arithmetic Shift Right (ASR)**: αριθμητική ολίσθηση προς τα δεξιά – **συμπλήρωση με ψηφία ίδια με το πρόσημο**
- **Rotate Right (ROR)**: κυκλική ολίσθηση προς τα δεξιά

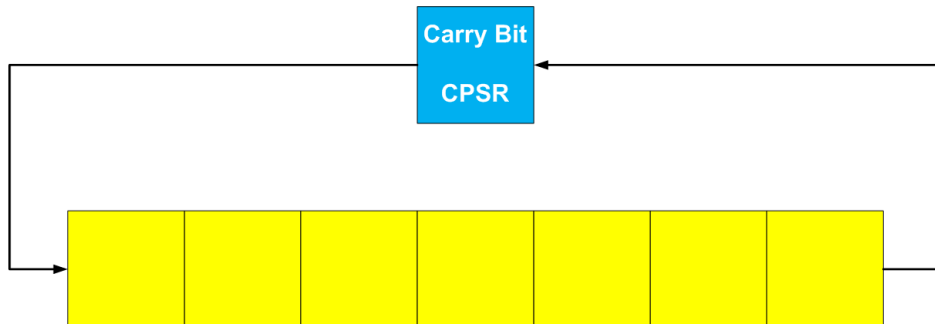
21

## Ολίσθηση προς τα αριστερά κατά 2 θέσεις



22

## Κυκλική ολίσθηση προς τα δεξιά



- Παράδειγμα : 1001\_0111 & Carry Bit = **X**
- Κυκλική δεξιά ολίσθηση κατά 3 θέσεις
- Νέα τιμή : **11X1\_0010**
- Αριστερή κυκλική ολίσθηση n: δεξιά κυκλική ολίσθηση κατά 33-n

23

## Εντολές Μεταφοράς Δεδομένων

- **LDR** (μεταφορά από μνήμη σε καταχωρητή):  $R \leftarrow M$
- **STR** (μεταφορά από καταχωρητή σε μνήμη):  $R \rightarrow M$



24

## Εντολές Μεταφοράς Δεδομένων

- **MOV** (μεταφορά από καταχωρητή σε καταχωρητή):  $R \leftarrow R$
- **MVN** (μεταφορά από καταχωρητή σε καταχωρητή με αντιστροφή):  $R \leftarrow \text{NOT}(R)$



- **SWP** (εναλλαγή τιμής από μνήμη σε καταχωρητή):  $R \leftrightarrow M$



25

## Πρόσθετες Εντολές Μεταφοράς Δεδομένων

- **LDM** (μεταφορά block από μνήμη σε καταχωρητές):  $\text{regs} \leftarrow \text{mem block}$
- **STM** (μεταφορά block από καταχωρητές σε μνήμη):  $\text{regs} \rightarrow \text{mem block}$
- **MRS** (μεταφορά από τον CPSR σε καταχωρητή):  $\text{CPSR} \rightarrow R$
- **MSR** (μεταφορά από καταχωρητή στον CPSR):  $\text{CPSR} \leftarrow R$
- **MRC** (μεταφορά από συνεπεξεργαστή σε καταχωρητή):  $\text{Cop} \rightarrow R$
- **MCR** (μεταφορά από καταχωρητή σε συνεπεξεργαστή):  $\text{Cop} \leftarrow R$

26

## Εύρος Μεταφοράς Δεδομένων

- Κάθε μεταφορά μπορεί να γίνει με δεδομένα εύρους :
  - 32-bit (word)
  - 16-bit (halfword)
  - 8-bit (byte)
- Μεταφορά 16-bit: **επίθεμα H** (halfword)
  - LDR → LDRH
- Μεταφορά 8-bit: **επίθεμα B** (byte)
  - LDR → LDRB

27

## Εντολές Μεταφοράς Δεδομένων: Σύνταξη

**<εντολή> Rd,[Rb,<offset>]**

- Το <offset> μπορεί να είναι καταχωρητής ή 8-bit τιμή με ολίσθηση 0 έως 31 ψηφίων
- Η παραπάνω σύνταξη βοηθάει στην διαχείριση πινάκων:
  - Rb είναι η αρχή του πίνακα
  - <offset> δείχνει στην τρέχουσα θέση του πίνακα
- Προσοχή στο μέγεθος των περιεχομένων κάθε θέσης π.χ.:
  - Αν είναι 1 byte αύξηση της διεύθυνσης κατά 1
  - Αν είναι word (4 bytes) αύξηση της διεύθυνσης κατά 4
- Οι θέσεις πρέπει να είναι στοιχισμένες σε διευθύνσεις που είναι πολλαπλάσια του 1 ή 4 αντίστοιχα (byte, halfword ή word alignment)

28

## Αποθήκευση αριθμού μεγαλύτερου του ενός byte στη μνήμη

- **Little Endian:** Το λιγότερο σημαντικό ψηφίο αποθηκεύεται στην χαμηλότερη διεύθυνση.
- **Big Endian:** Το περισσότερο σημαντικό ψηφίο αποθηκεύεται στην χαμηλότερη διεύθυνση.
- Στον AT91 επιτρέπονται και οι δύο τρόποι αλλά το Linux υιοθετεί Little Endian και αυτή την σύμβαση θα πρέπει να ακολουθούν όλα τα προγράμματα.

29

## Παράδειγμα:

NUMBER	0xDEADBEEF	
ΘΕΣΗ ΜΝΗΜΗΣ	LITTLE ENDIAN	BIG ENDIAN
0xC000450	EF	DE
0xC000451	BE	AD
0xC000452	AD	BE
0xC000453	DE	EF

30

## Προσπέλαση διαδοχικών θέσεων πινάκων με τροποποίηση καταχωρητή βάσης

- Τροποποίηση πριν την προσπέλαση
  - `LDR R5,[R0,R2, LSL #2]!`
  - Γίνεται αύξηση του `R0` κατά  $R2 * 4$ , μετά ο `R0` γίνεται  $R0 + (R2 * 4)$  και μετά θα διευθυνσιοδοτηθεί η κύρια μνήμη.
- Τροποποίηση μετά την προσπέλαση
  - `LDR R5,[R0],R2,LSL #2`
  - Γίνεται προσπέλαση στην διεύθυνση `R0` και μετά ο `R0` γίνεται ίσος με  $R0 + (R2 * 4)$

31

## Παραδείγματα φόρτωσης:

### Load - LDR

Υποθέστε ότι τα περιεχόμενα έχουν ως εξής: `R1` `0x8000` και `R5` `0x6`

Εντολή	Περιγραφή
<code>LDR R0, [R1]</code>	Μεταφέρει το περιεχόμενο της θέσης μνήμης με διεύθυνση <code>0x8000</code> στον καταχωρητή <code>R0</code> .
<code>LDR R0, [R1,R5]</code>	Μεταφέρει το περιεχόμενο της θέσης μνήμης με διεύθυνση $0x8000 + 6 = 0x8006$ στον καταχωρητή <code>R0</code> .
<code>LDR R0 [R1, #10]</code>	Μεταφέρει το περιεχόμενο της θέσης μνήμης με διεύθυνση $0x8000 + 0xA = 0x800A$ στον καταχωρητή <code>R0</code> .
<code>LDR R0, [R1, R5, LSL, #2]</code>	Μεταφέρει το περιεχόμενο της θέσης μνήμης με διεύθυνση $0x8000 + 6 * 4 = 0x8018$ στον καταχωρητή <code>R0</code> .
<code>LDR R0, [R1, R5, LSL, #4]!</code>	Μεταφέρει το περιεχόμενο της θέσης μνήμης με διεύθυνση $0x8000 + 6 * 16 = 0x8060$ στον καταχωρητή <code>R0</code> και αποθηκεύει την τιμή <code>0x8060</code> στον καταχωρητή <code>R1</code> .
<code>LDR R0, [R1], #12</code>	Μεταφέρει το περιεχόμενο της θέσης μνήμης με διεύθυνση <code>0x8000</code> στον καταχωρητή <code>R0</code> και αποθηκεύει την τιμή $0x8000 + 12 = 0x800C$ στον καταχωρητή <code>R1</code> .

32



## Παραδείγματα αποθήκευσης:

### Store - STR

Υποθέστε ότι τα περιεχόμενα έχουν ως εξής:

**R1** **0x8000**

και

**R5** **0x6**

Εντολή	Περιγραφή
<b>STR R0, [R1]</b>	Μεταφέρει στη θέση μνήμης με διεύθυνση 0x8000 το περιεχόμενο του καταχωρητή R0.
<b>STR R0, [R1, #20]</b>	Μεταφέρει στη θέση μνήμης με διεύθυνση $0x8000 + 20 = 0x8014$ το περιεχόμενο του καταχωρητή R0.

33

## Εντολές μεταφοράς μπλοκ καταχωρητών

**<εντολή> <mode> Rb, {<regs>}**

- Μεταφέρουν πολλαπλές τιμές από/προς μνήμη προς/από σύνολο καταχωρητών που δηλώνονται μέσα σε άγκιστρα χωρισμένοι με κόμμα, ή συνδέονται με παύλα (συνεχόμενοι).
- **Rb**: περιέχει την αρχική διεύθυνση της κύριας μνήμης.
- **Mode**:
  - Increment Before (IB)
  - Increment After (IA)
  - Decrement Before (DB) Decrement After (DA)

34

## Παραδείγματα

Το περιεχόμενο του R1 είναι **R1 0x8000**

Εντολή	Περιγραφή
<b>LDMIB R1, {R0, R2-R5}</b>	Θα ξεκινήσει από την θέση 0x8004 και θα τοποθετήσει το περιεχόμενό της στον R0, το περιεχόμενο της 0x8008 στον R2 κλπ.
<b>LDMDA R1, {R0, R2-R5}</b>	Θα ξεκινήσει από την θέση 0x8000 και θα τοποθετήσει το περιεχόμενό της στον R5, το περιεχόμενο της 0x7FFC στον R4 και τέλος το περιεχόμενο της 0x7FF0 στον R0.
<b>LDMDA R1!, {R0, R2-R5}</b>	Θα εκτελέσει την ίδια διαδικασία με την προηγούμενη εντολή, αλλά τώρα η τελευταία υπολογισθείσα διεύθυνση θα αποθηκευτεί στον R1 ( $0x8000 - 4 * 5 = 0x7FEC$ ).
<b>STMIA R1, {R0, R2-R5}</b>	Θα τοποθετήσει στο 0x8000 το περιεχόμενο του R0, στο 0x8004 το περιεχόμενο του R2 κλπ.
<b>STMDB R1, {R0, R2-R5}</b>	Θα τοποθετήσει το περιεχόμενο του R5 στο 0x7FFC, το περιεχόμενο του R4 στο 0x7FF8 κλπ.

35

## Μεταφορά καταχωρητή σε καταχωρητή

**MOV Rd,<operand>**

- <operand>: καταχωρητής ή τιμή **8-bit με ολίσθηση**
- Δεν είναι δυνατή η άμεση φόρτωση καταχωρητή με οποιαδήποτε τιμή
- Μεταφορές ειδικού τύπου όπως στο συνεπεξεργαστή ή από τον CPSR

36

## Παραδείγματα

Εντολή	Περιγραφή
MOV R1, R2	Μεταφέρει στον R1 το περιεχόμενο του R2.
MOV R1, R2, LSL #5	Μεταφέρει στον R1 το περιεχόμενο του R2, ολισθημένο αριστερά κατά 5 θέσεις.
MOV <sup>S</sup> R1, R2, LSL #5	Εκτελεί την ίδια διαδικασία με την προηγούμενη εντολή, αλλά τώρα αποθηκεύεται στο Carry bit του CPSR το τελευταίο bit που ολίσθησε εκτός του καταχωρητή.
MOV R1, #0x35	Μεταφέρει στον R1 την τιμή 0x35.
MVN R1, #0x35	Μεταφέρει στον R1 την τιμή #0xFFF_FCA.

37

## ΑΡΙΘΜΗΤΙΚΕΣ – ΛΟΓΙΚΕΣ ΠΡΑΞΕΙΣ

### Ενότητα 3

38

## Παραδείγματα: Αριθμητικές & Λογικές Πράξεις

Εντολή	Περιγραφή
ADD R1, R1, #0x10	Προσθέτει στον καταχωρητή R1 την τιμή 0x10.
SUB R5, R1, R2, LSL #2	Αφαιρεί από τον R1 το $R2 \times 4$ και αποθηκεύει το αποτέλεσμα στον R5.
RSB R5, R1, R2, LSL #2	Αφαιρεί από το $R2 \times 4$ τον R1 και αποθηκεύει το αποτέλεσμα στον R5.
ADC R5, R1, R2	Προσθέτει τον R1 με τον R2 συνυπολογίζοντας και τη σημαία κρατούμενου του CPSR και αποθηκεύει το αποτέλεσμα στον R5.
AND R1, R1, #0x10	Εκτελεί λογική πράξη AND ανάμεσα στον R1 και την άμεση τιμή 0x10 και αποθηκεύει το αποτέλεσμα στον R1.

39

## Παραδείγματα Πράξεων: Συγκρίσεις

Εντολή	Περιγραφή
CMP R2, R4 R2-R4 (flags affected)	Συγκρίνει το περιεχόμενο του R2 με το περιεχόμενο του R4. Ανάλογα με το είδος του αποτελέσματος ανανεώνονται οι σημαίες κατάστασης.
CMP R3, 0x8000 R3-0x8000 (flags affected)	Συγκρίνει το περιεχόμενο του R3 με το δεδομένο 0x8000. Ανάλογα με το είδος του αποτελέσματος ανανεώνονται οι σημαίες κατάστασης.
CMN R1, R2 R1-(-R2) (flags not affected)	Συγκρίνει το περιεχόμενο του R1 με το -R2 (προσθέτει το συμπλήρωμα ως προς 2 του R2) και δεν επηρεάζει τις σημαίες κατάστασης.
TEQS R1, R4 R1 XOR R4, R1 = (?) R4 (Zero Bit affected)	Συγκρίνει ως προς την ισότητα το περιεχόμενο του R1 με το περιεχόμενο του R4. Αν τα περιεχόμενα των καταχωρητών είναι ίδια, η σημαία μηδενικού αποτελέσματος του CPSR γίνεται 1.

40

## Παραδείγματα Αριθμητικών-Λογικών Πράξεων

Εντολή	Περιγραφή
<b>MUL R1, R2, R5</b> $R1 \leftarrow 32 \text{ LSBs of } R2 * R5$	Πολλαπλασιάζει το περιεχόμενο του <b>R2</b> με αυτό του <b>R5</b> και αποθηκεύει τα 32 λιγότερο σημαντικά bits στον καταχωρητή <b>R1</b> .
<b>MLA R1, R1, R4, R6</b> $R1 \leftarrow 32 \text{ LSBs of } R6 + R1 * R4$	Πολλαπλασιάζει το περιεχόμενο του <b>R1</b> με αυτό του <b>R4</b> , προσθέτει στο γινόμενο το περιεχόμενο του <b>R6</b> και αποθηκεύει τα 32 λιγότερα bits στον καταχωρητή <b>R1</b> .
<b>SMULL R0, R1, R4, R6</b> $R1: R0 \leftarrow R4 * R6$	Πολλαπλασιάζει το περιεχόμενο του <b>R4</b> με αυτό του <b>R6</b> με προσημασμένη αριθμητική και αποθηκεύει τα 32 λιγότερο σημαντικά bits στον καταχωρητή <b>R0</b> και τα 32 περισσότερα σημαντικά bits στον <b>R1</b> .
<b>UMLAL R1, R2, R4, R6</b> $R2: R1 \leftarrow R4 * R6 (R1 + R2 * 2^{32})$	Πολλαπλασιάζει το περιεχόμενο του <b>R4</b> με αυτό του <b>R6</b> με μη προσημασμένη αριθμητική, προσθέτει στο γινόμενο την τιμή $R1 + R2 * 2^{32}$ και αποθηκεύει τα 32 λιγότερο σημαντικά bits στον καταχωρητή <b>R1</b> και τα 32 περισσότερα σημαντικά bits στον <b>R2</b> .

41

## Εντολές Διακλάδωσης

Εντολή	Περιγραφή
<b>B</b>	Απλή διακλάδωση
<b>BL</b>	Διακλάδωση με αποθήκευση διεύθυνσης επιστροφής από τον R15 στο R14 (για κλήση υπορουτίνας)
<b>BX, BLX</b>	Όμοια με B, BL αντίστοιχα αλλά και αλλαγή συνόλου εντολών (από ARM σε Thumb)

- Παραδείγματα:  
 $B \text{ Loop};$  Άλμα στην ετικέτα *Loop*  
 $BLEQ \text{ Loop};$  Άλμα στην ετικέτα *Loop* αν  $Z=1$

42

# AT91 : ASSEMBLY

## Ενότητα 4

43

## Σύνταξη Εντολών

[ετικέτα:] <κενο> εντολή <κενό> δεδομένα @ σχόλια

### Directives:

- .abort** Σταματά τη μετάφραση
- .align** Τοποθέτηση δεδομένων σε στοιχισμένες θέσεις
- .arm** Έναρξη 32-bit κώδικα (arm)
- .asciz** Αρχή ακολουθίας αλφαριθμητικών χαρακτήρων
- .byte** Αρχή ακολουθίας bytes
- .data** Ακολουθούν δεδομένα
- .equ** Αντιστοίχιση τιμής σε σύμβολο
- .global** Δηλώνει ένα σύμβολο ως σφαιρικό
- .hword** Τοποθετεί halfwords
- .if** Ο κώδικας που ακολουθεί εκτελείται αν ισχύει η συνθήκη του
- .include** Εισάγει τον κώδικα άλλου αρχείου στην τρέχουσα θέση
- .text** Ακολουθεί κώδικας
- .word** Τοποθετεί words

44

## Παραδείγματα directives

**.align 4** @Τα δεδομένα που ακολουθούν να είναι στοιχισμένα σε διευθύνσεις που είναι πολλαπλάσια του 4

**Label:**

**.asciz "Test"** @ Οι χαρακτήρες της λέξης Test θα αποθηκευθούν σε διαδοχικές θέσεις μετά τη διεύθυνση της Label

**Label:**

**.byte 0x10, 0x20, 0x30, 0x33** @οι 4 τιμές μεγέθους byte θα αποθηκευθούν σε διαδοχικές θέσεις μετά τη διεύθυνση Label

**.equ Mac, #0x1234** @ όπου συναντάται η λέξη Mac θα αντικαθίσταται από την άμεση τιμή 0x1234

**.global main** @ η ετικέτα main θα μπορεί να προσπελαστεί από άλλα προγράμματα

**.hword 0x2010, 0x3330** @ όμοια με .byte παραπάνω

**.word 0x33302010** @ όμοια με .byte / .hword παραπάνω

45

## Παραδείγματα directives

**.equ CFG, 1**

**.if CFG**

**MOV R0, #0x30**

**.else**

**MOV R0, #0x10**

**.endif**

**.include "sfina.asm"**

46

# ΕΞΟΜΟΙΩΤΗΣ KEIL

## Ενότητα 5

47

## Ρυθμίσεις – Εγκατάσταση Εργαλείων

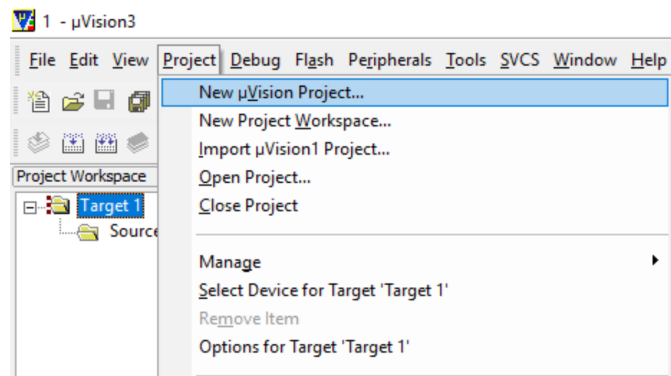
- Εγκατάσταση GNU εργαλείων:
  - Θα βρείτε τα κατάλληλα αρχεία για την εγκατάσταση του εξομοιωτή στο διαδικτυακό τόπο:
    - *KEIL Microcontroller Development Tools*, [www.keil.com](http://www.keil.com)
    - *GNU Tools*, διαθέσιμο και στην ηλεκτρονική τάξη του μαθήματος
  - Τα αρχεία που θα χρειαστούμε είναι τα ακόλουθα:
  - Simulator (Keil's mVision, GNU Tools).
  - Πριν γίνει η εγκατάσταση του εξομοιωτή πρέπει να εγκαταστήσουμε τον κατάλογο C:\Cygpus, τρέχοντας το gccarm331.exe.
  - Έπειτα τρέχουμε την αντίστοιχη τελευταία έκδοσης εφαρμογή (KEIL MDT), για την εγκατάσταση του KEIL.

48



## Δημιουργία Project

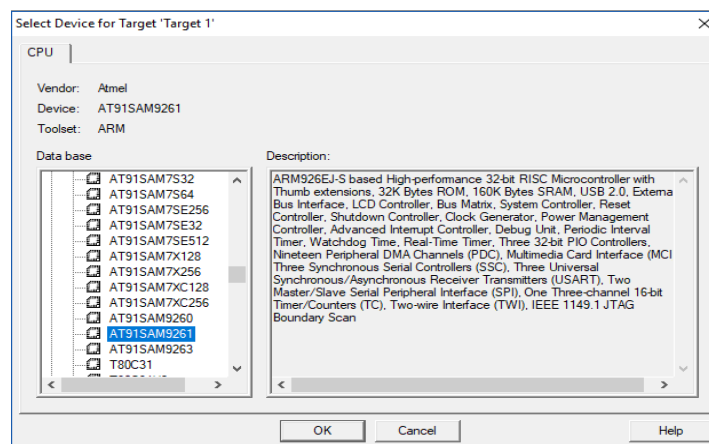
- Project → New uVision project → όνομα
- Επιλογή επεξεργαστή από Project → Select Device for target 'Target 1'



49

## Επιλογή Device: Target 1

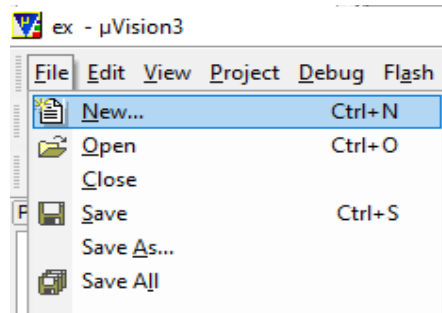
- Επιλέγουμε την εταιρία Atmel και από τους επεξεργαστές που μας παρέχονται διαλέγουμε τον AT91SAM9261



50

## Δημιουργία Νέου αρχείου

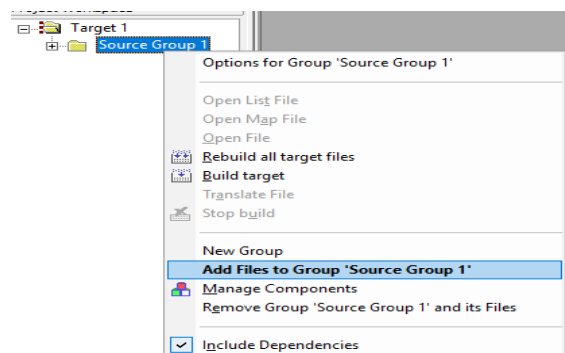
- Επιλέγουμε από το menu File το New , και στο παράθυρο που εμφανίζεται πληκτρολογούμε τον κώδικα μας.
- Για την αποθήκευση του project, διαλέγουμε το Save As από το menu File και δίνουμε στο αρχείο μας ένα όνομα με κατάληξη .s



51

## Δημιουργία Νέου αρχείου

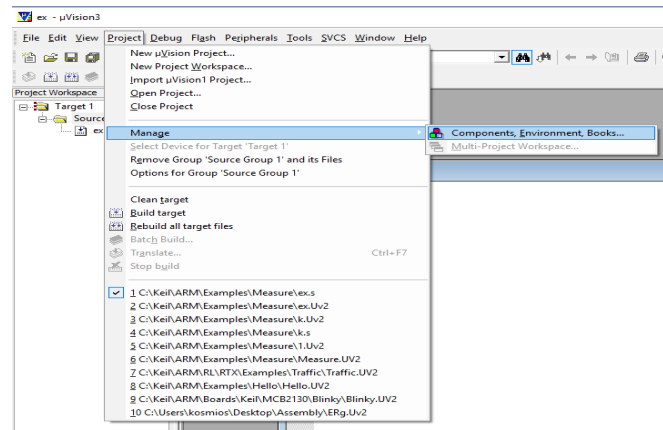
- Επιλέγουμε στο Project workspace το Target 1 κ' με δεξί κλικ στο source Group 1 επιλέγουμε Add Files to Group, και διαλέγουμε το αρχείο που μόλις αποθηκεύσαμε.



52

## Ρυθμίσεις Εξομοιωτή

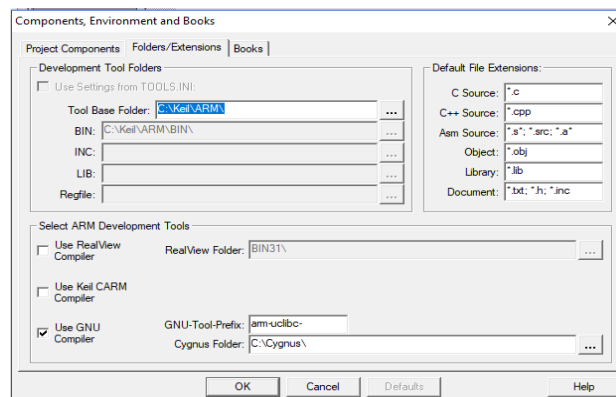
1. Project → Manage → Components, Environment and Books...



53

## Ρυθμίσεις Εξομοιωτή

2. Μετάβαση σε καρτέλα Folders/Extensions
  - GNU tool prefix: arm-uclibc-
  - Cygnus Folder: C:\Cygnus\

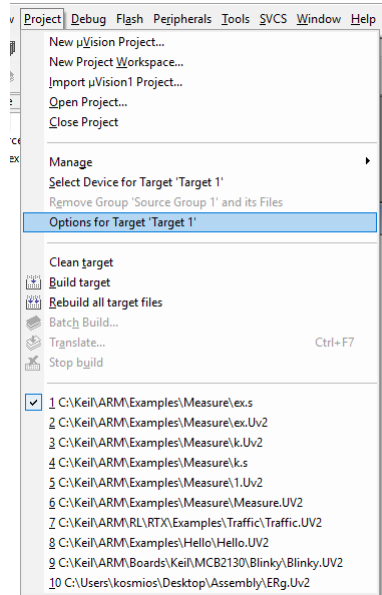


54

## Ρυθμίσεις Εξομοιωτή

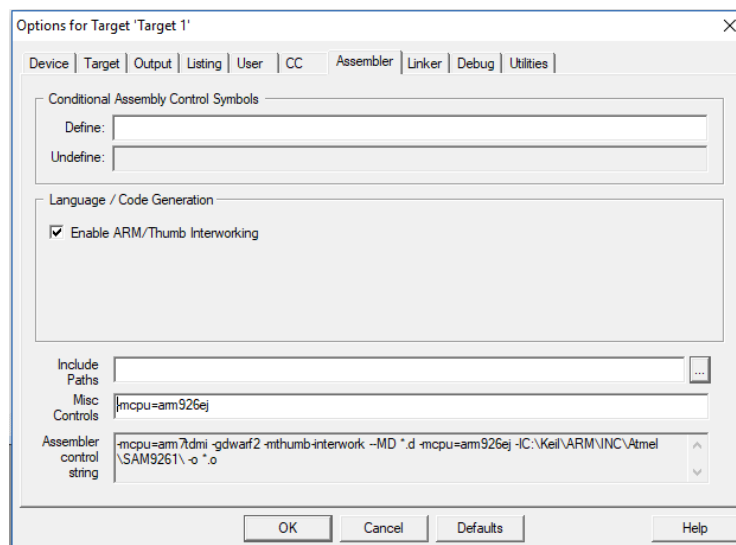
### 3. Project → Options for Target 1 → Assembler

- Misc Controls: -mcpu=arm926ej  
 (αυτό το option θα δηλώσει τον σωστό επεξεργαστικό πυρήνα και θα παρακάμψει το ARM7TDMI που διαλέξαμε αναγκαστικά προηγουμένως)



55

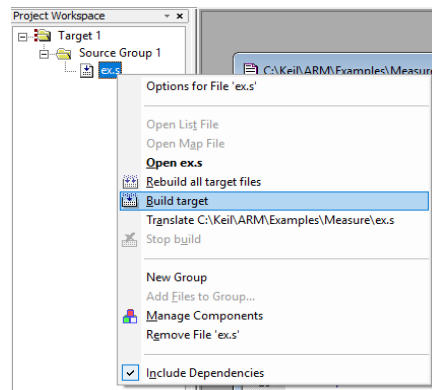
## Ρυθμίσεις Assembler



56

## Σύνδεση Αρχείου με το Project

- Μετάφραση κώδικα με **Project→Build Target.**
- Μετά την μετάφραση του κώδικα θα σας εμφανίσει μηνύματα στο κάτω μέρος της κονσόλας.



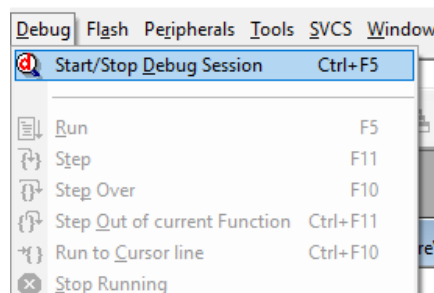
```

> Build target 'Target 1'
> assembling ex.s...
> linking...
> /cygdrive/c/Cygnus/Arm-Tools/Bin/./lib/gcc-lib/arm-thumb-elf/3.3.1/./././././arm-thumb-elf/bin/ld: warning:
> "ex.elf" - 0 Error(s), 0 Warning(s).

```

57

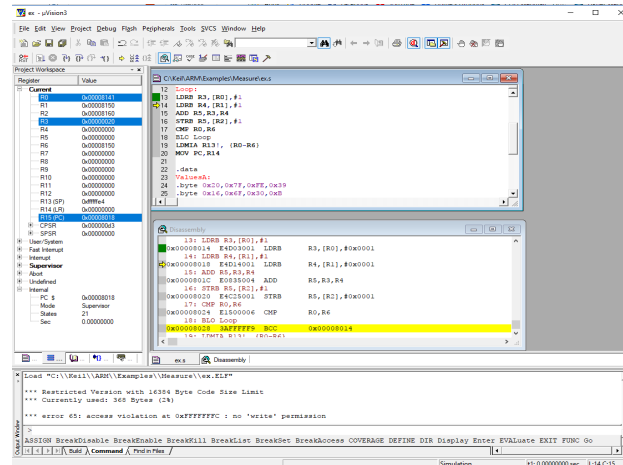
Εκτέλεση Πηγαίου Κώδικα:



- Με την ολοκλήρωση της προηγούμενης διαδικασίας συνεχίζουμε με το debugging διαλέγοντας **Start/Stop debug session** από το Debug menu.

58

## Εκτέλεση Αρχείου Κώδικα:



- Για την εκτέλεση του κώδικά, μπορούμε να θέσουμε στον PC την τιμή 0x8000.

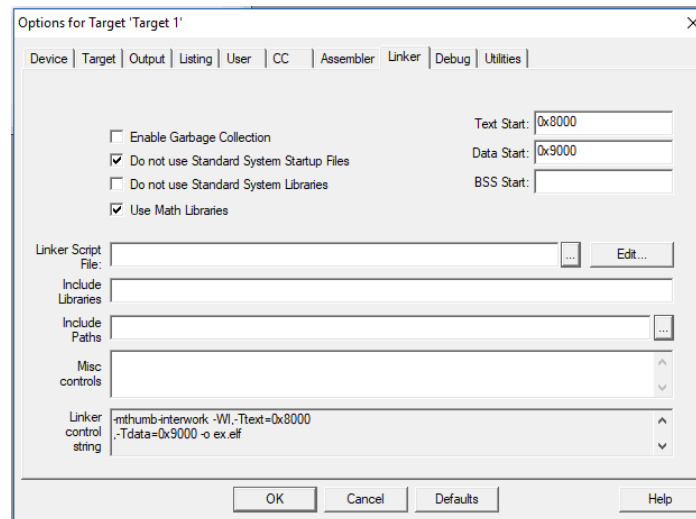
59

## Προσοχή κατά την εξομοίωση με Keil

- Αν στον κώδικά μας χρησιμοποιείται στοίβα ή κλήση υπορουτινών ο δείκτης στοίβας (SP-R13) θα πρέπει να οριστεί στην κορυφή της περιοχής στοίβας
- Επειδή ζητήσαμε άλλο μοντέλο επεξεργαστή από το αναπτυξιακό του εργαστηρίου δεν συμφωνούν απόλυτα οι διευθύνσεις εσωτερικής μνήμης.
- Αυτό θα μπορούσε να διορθωθεί από την επόμενη καρτέλα αλλά δεν είναι απαραίτητο μιας και τις διευθύνσεις που χρησιμοποιούμε τις δηλώνουμε με ονόματα (labels κι όχι με νούμερα) στο .data segment.

60

## Ρυθμίσεις...



61

## Εκτέλεση κώδικα στο περιβάλλον του αναπτυξιακού AT91

- Συγγραφή αρχείου assembly με τη βοήθεια του editor (πχ, **edit test.s**)
- Εναλλακτικά μπορεί να χρησιμοποιηθεί κάποιο έτοιμο αρχείο αποθηκευμένο σε ένα flash stick:
  - `mount /dev/sda /mnt` ή
  - `mount /dev/sda1 /mnt` ή
  - `mount /dev/sda2 /mnt`
- Το κατάλληλο format που υποστηρίζει η πλατφόρμα του AT91 είναι "NTFS".

62

## Μετάφραση κώδικα assembly

- `make as test.s test`
- Εκκίνηση GNU debugger:
  - `gdb test`
- Αν η gdb παρουσιάζεται ως άγνωστη εντολή
  - `cd /storage/.gnu`
  - `chmod 777 gdb`
  - Κλήση της ως `/storage/.gnu/gdb test`

63

## Εντολές GDB: Συχνή Χρήση

Εντολή	Περιγραφή
<code>break main</code>	Τοποθετεί το breakpoint στη θέση που χαρακτηρίζεται από την ετικέτα <code>main</code> .
<code>break Check</code>	Τοποθετεί το breakpoint στη θέση που χαρακτηρίζεται από την ετικέτα <code>Check</code> .
<code>break *0x80c0</code>	Τοποθετεί το breakpoint στην διεύθυνση <code>0x80c0</code> της μνήμης.
<code>info break</code>	Προβάλλει τη λίστα με όλα τα breakpoints που έχουν τοποθετηθεί.
<code>delete 2</code>	Διαγράφει το δεύτερο breakpoint.
<code>clear 12</code>	Διαγράφει το breakpoint που έχει τεθεί στη γραμμή 12.
<code>run</code>	Το πρόγραμμα τρέχει μέχρι το επόμενο breakpoint.
<code>s</code>	Εκτελεί την επόμενη εντολή (step).
<code>c</code>	Συνεχίζει την κανονική εκτέλεση μετά από διακοπή (continue).

64



## Περιεχόμενα μνήμης/καταχωρητών

- Εντολή: **x/nfu <address>**
  - **n**: καθορίζει το **πλήθος** των **διαδοχικών τιμών που θα προβληθούν**
  - **f**: καθορίζει τον **τρόπο προβολής**. Παράδειγμα:
    - x : δεκαεξαδική μορφή (hex)
    - u: μη-προσημασμένη δεκαδική μορφή (dec)
    - t: δυαδική μορφή (bin)
  - **u**: καθορίζει το **μέγεθος των τιμών**
    - b (byte)
    - h (halfword)
    - w (word)
    - g (giantword)

65

## Παραδείγματα Περιεχομένων μνήμης/καταχωρητών

Εντολή	Περιγραφή
<b>x/4xb 0x10544</b>	Θα εμφανίσει 4 bytes σε δεκαεξαδική μορφή ξεκινώντας από τη διεύθυνση 0x10544.
<b>x/8tw (&amp;Values+2)</b>	Θα εμφανίσει 8 words σε δυαδική μορφή ξεκινώντας από τη διεύθυνση of Values+2.
<b>x/dw &amp;Values</b>	Θα εμφανίσει την δεκαδική τιμή του word Values.

Εντολή	Περιγραφή
<b>p/x *((char*) &amp;Values+2)</b>	Θα εμφανίσει την byte τιμή σε δεκαεξαδική μορφή της διεύθυνσης Values+2.
<b>p/x *((short*) &amp;Values)</b>	Θα εμφανίσει την halfword τιμή σε δεκαεξαδική μορφή της διεύθυνσης Values.
<b>p/x *((long*) &amp;Values)</b>	Θα εμφανίσει την word τιμή σε δεξαεξαδική μορφή της διεύθυνσης Values.
<b>p/x \$r2</b>	Θα εμφανίσει το περιεχόμενο του καταχωρητή R2.

66

## Άλλες εντολές του GDB

Εντολή	Περιγραφή
<code>list</code>	Θα εμφανιστούν 10 γραμμές κώδικα γύρω από την γραμμή που εισάγαμε.
<code>list 15,18</code>	Θα εμφανιστούν οι γραμμές 15 έως 18.
<code>disassemble (main+20)</code>	Θα εμφανιστούν οι εντολές από τη θέση <code>main</code> έως τη θέση <code>main + 40</code> (κάθε εντολή καταλαμβάνει 4 bytes).

67

## ΒΙΒΛΙΟΓΡΑΦΙΑ

Ενότητα 6

68

## || Βιβλιογραφία - Αναφορές:

- Χ. Βέργος, «Εγχειρίδιο Χρήσης AT91», Τμήμα Εκτυπώσεων – Τυπογραφείο Πανεπιστημίου Πατρών, 2017-18.
- Δημήτριος Νικολός, «Αρχιτεκτονική Υπολογιστών», 2<sup>η</sup> Έκδοση, 2012, ISBN: 978-960-93-4168-4.
- KEIL, Microcontroller Development Tools, διαθέσιμα στο διαδικτυακό τόπο: [www.keil.com](http://www.keil.com)