

HPY 201– Ψηφιακοί Υπολογιστές

CLANG: Chania C-Level Assembly Language (ή
«Intermediate Code»)

Μοντέλο προγραμματισμού MIPS

- Οι λέξεις (words) έχουν πλάτος 32 bits
- 32 καταχωρητές
- Ο κάθε καταχωρητής έχει πλάτος 32 bits
- Η πρόσβαση στη μνήμη γίνεται με χρήση των εντολών load, store.
- Όλες οι αριθμητικές και λογικές πράξεις αποθηκεύουν το αποτέλεσμα τους σε καταχωρητή
- Η εκτέλεση των εντολών γίνεται στην σειρά, εκτός εάν η ροή του προγράμματος αλλάξει λόγω εντολής ελέγχου ροής

CLANG

- Η CLANG είναι μία ενδιάμεση “γλώσσα” προγραμματισμού
- Στόχος μαθήματος είναι η εξεικοίωση με την γλώσσα μηχανής (assembly MIPS)
- C ➔ CLANG ➔ Assembly MIPS

Δήλωση Μεταβλητών και Συναρτήσεων

- Οι δηλώσεις και οι αρχικοποιήσεις των μεταβλητών καθώς και οι δηλώσεις των συναρτήσεων γίνονται όπως ακριβώς στην C.
- **ΠΡΟΣΟΧΗ!** Η κάθε μεταβλητή αποτελεί χώρο που έχει δεσμευτεί στην μνήμη.
- Παράδειγμα:

C	CLANG
<pre>int main() { int x = 5; } int foo() { }</pre>	<pre>Int R0 = 0, R1, R2, R3, R4, int main() { int x = 5; } int foo() { }</pre>

Επεξεργασία (I)

- Οι υπολογισμοί δεν γίνονται με την βοήθεια των μεταβλητών αλλά μόνο των καταχωρητών.
- **ΠΡΟΣΟΧΗ!** Πριν την επεξεργασία μίας μεταβλητής θα πρέπει πρώτα η τιμή της να “περνάει” σε καταχωρητή και στην συνέχεια αφού γίνει η επεξεργασία να ξαναγυρίζει στην μεταβλητή.
- Υπάρχουν 32 καταχωρητές:
 - $R_i, 0 \leq i \leq 31,$
 - **Σύμβαση:** Ο καταχωρητής R0 θα έχει πάντα την τιμή 0.

Επεξεργασία (II)

Υπολογισμοί:

$R_i = \text{var};$ $/*\ 0 \leq i \leq 31, \text{var} = \text{μεταβλητή της } C\ */$

$\text{var} = R_i;$ $/*\ 0 \leq i \leq 31, \text{var} = \text{μεταβλητή της } C\ */$

$R_i = R_j \text{ op } R_k;$ $/*\ 0 \leq i, j, k \leq 31, \text{op} = \text{τελεστής της } C\ */$

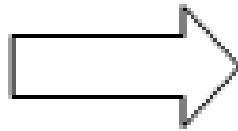
$R_i = R_j \text{ op } \text{const};$ $/*\ 0 \leq i, j, k \leq 31, \text{op} = \text{τελεστής της } C, \text{const} = \text{σταθερή}\ */$

- Παραδείγματα:

C	CLANG
<pre>int x = 5; x = x + 6;</pre>	<pre>int x = 5; R1 = x; R1 = R1 + 6; x = R1;</pre>
<pre>int x = 3, y = 7; int z; z = y - x;</pre>	<pre>int x = 3, y = 7; int z; R1 = x; R2 = y; R3 = R2 - R1; z = R3;</pre>

Επεξεργασία Μεταβλητών

```
int x, y, z;  
...  
x = y * z + 3;  
y = -1;
```



```
int x, y, z;  
...  
R3 = y;  
R4 = z;  
R8 = R3 * R4;  
R8 = R8 + 3;  
x = R8;  
R1 = -1;  
y = R1;
```

Έλεγχος Ροής (if)

- Χρήση Labels στην επόμενη εντολή από το if ή στην θέση του else
 - Τα Labels “κρύβουν” διευθύνσεις μνήμης όπου είναι αποθηκευμένη μία εντολή του προγράμματος
- Χρήση if με την αντίστροφη συνθήκη που περιγράφεται στο πρόγραμμα.
- Χρήση συνθήκης όπως στην C (<, <=, ...)
- Χρήση goto για μετάβαση στην εντολή που έχει τοποθετηθεί το label.

C	CLANG
<pre>int y = 5, x = 3, z; if(x > 3) { z = y + x; } z = z - 1;</pre>	<pre>int y = 5, x = 3, z; R1 = y; R2 = x; R3 = 3; if(R2 <= R3) goto after_if_label; R4 = R1 + R2; after_if_label: R4 = R4 - 1;</pre>

Έλεγχος Ροής (if-else)

- Παραδείγματα

C	CLANG
<pre>int y = 5, x = 3, z; If(x > 3) { z = y + x; } Else { z = y - x; } z = z - 1;</pre>	<pre>int y = 5, x = 3, z; R1 = y; R2 = x; R3 = 3; If(R2 <= R3) goto else_label; R4 = R1 + R2; goto after_cond; else_label: R4 = R1 - R2; After_cond: R4 = R4 - 1;</pre>

CLANG: Chania C-Level Assembly Language (η «Intermediate Code»)

Αντιστροφή της συνθήκης!!!

```
if (x < y) {  
    s1;  
} else {  
    s2;  
}
```



```
R3 = x;  
R4 = y;  
if !(R3 < R4) goto lab_else  
/* κώδικας CLANG για το s1 */  
goto lab_endif;  
lab_else:  
/* κώδικας CLANG για το s2 */  
lab_endif:
```

If – Else if – Else?

C

```
int y = 5, x = 3, z;  
If(x > 3)  
{  
    z = y + x;  
}  
Else if (x == 3)  
{  
    z = y - x;  
}  
Else  
{  
    z = 5;  
}  
z = z - 1;
```

CLANG

```
int y = 5, x = 3, z;  
R1 = y;  
R2 = x;  
R3 = 3;  
  
??
```

If – Else if – Else: Μετατροπή

C

```
int y = 5, x = 3, z;  
If(x > 3)  
{  
    z = y + x;  
}  
Else  
{  
    if (x == 3)  
    {  
        z = y - x;  
    }  
    Else  
    {  
        z = 5;  
    }  
}  
z = z - 1;
```

CLANG

```
int y = 5, x = 3, z;  
R1 = y;  
R2 = x;  
R3 = 3;  
  
If( R2 <= R3) goto else_label;  
R4 = R1 + R2;  
goto after_cond;  
  
else_label:  
.....  
  
After_cond:  
R4 = R4 - 1;
```

If – Else if – Else: Μετατροπή

C

```
int y = 5, x = 3, z;
```

```
If(x > 3)
```

```
{
```

```
    z = y + x;
```

```
}
```

```
Else
```

```
{
```

```
    if (x == 3)
```

```
    {
```

```
        z = y - x;
```

```
    }
```

```
    Else
```

```
    {
```

```
        z = 5;
```

```
    }
```

```
}
```

```
z = z - 1;
```

CLANG

```
int y = 5, x = 3, z;
```

```
R1 = y;
```

```
R2 = x;
```

```
R3 = 3;
```

```
If( R2 <= R3) goto else_label;
```

```
R4 = R1 + R2;
```

```
goto after_cond;
```

```
else_label:
```

```
If ( R2 != R3) goto else_label_2;
```

```
R4 = R1 - R2;
```

```
goto after_cond;
```

```
else_label_2:
```

```
R4 = 5;
```

```
after_cond:
```

```
R4 = R4 - 1;
```

Loops (I)

- Αντικατάσταση loops με χρήση if και εντολές goto
- **While loop:**
 - while_label
 - Αντιστροφή συνθήκης
 - Goto while_label
- **Παράδειγμα**

C (while)

```
int y = 0, z = 20;  
while(y < 3)  
{  
    z = z - y;  
    y++;  
}  
z = z - 1;
```

CLANG (while)

```
int y = 0, z = 20;  
R1 = y;  
R2 = z;  
R3 = 3;  
While_label:  
    If( R1 >= R3) goto after_loop;  
    R2 = R2 - R1;  
    R1 = R1 + 1;  
    Goto while_label;  
after_loop:  
    R2 = R2 - 1;
```

CLANG: Chania C-Level Assembly Language (η «Intermediate Code»)

```
while (R2 < R4)
{
    s1;
}
```



```
lab_while_loop:
    if !(R3 < R4) goto lab_endwhile
    /* κώδικας CLANG για το s1 */
    goto lab_while_loop;
lab_endwhile:
```

Αντιστροφή της συνθήκης!!!

Loops (II)

- for loop:
 - Μετατροπή σε while
 - Μετατροπή του while σε CLANG
- Παράδειγμα

C (for)	CLANG (while)
<pre>int y = 0, z = 20; for(i = 0; i < z; i++) { y = y + z; }</pre>	<pre>int y = 0, z = 20, i = 0; R2 = z; R3 = y; R1 = i; R1 = 0; Label_loop: If (R1 >= R2) goto after_loop; R3 = R3 + R2; R1 = R1 + 1; Goto Label_loop; after_loop:</pre>
C (while)	
<pre>int y = 0, z = 20, i; i = 0; While(i < z) { y = y + z; i++; } </pre>	

Loops (III)

- **Do while loop:**
 - Στην ουσία είναι while loop με λίγη διαφορετική δομή
- **Παράδειγμα**

C (do while)	CLANG (do while)
<pre>int y = 0, z = 20; do{ z = z - y; y++; } while(y < 3)</pre>	<pre>int y = 0, z = 20; R1 = y; R2 = z; R3 = 3; do_label: R2 = R2 - R1; R1 = R1 + 1; If(R1 >= R3) goto after_loop; goto do_label; after_loop:</pre>

Κλήση συνάρτησης

- Κλήση συνάρτησης με τον ίδιο τρόπο όπως στη C
 - Παράμετροι μόνο με τη χρήση καταχωρητών
 - Επιστρεφόμενη τιμή στον καταχωρητή R2 (\$v0, σύμβαση)
- Παράδειγμα

C (while)	CLANG
<pre>void main() { int a = 3, b = 4; int y = foo(a, b); } int foo(int x, int y) { int z; z = x + y; return(z); }</pre>	<pre>void main() { int a = 3, b = 4; R1 = a; R3 = b; R2 = foo(R1, R3); y = R2; } void foo(int R1, int R3) { R2 = R1 + R3; return(R2); }</pre>

Θέματα Μνήμης

- C: Array 1D → CLANG: Global Array 1D
- C: Array 2D → CLANG: Global Array 1D (Μετατροπή του 2D όπως σας παρουσιάστηκε στο προηγούμενο μάθημα)
- **Παραδείγματα**

- `char str [8];`

6000

'H'	'e'	'l'	'l'	'o'	'\0'		
str [0]	[1]	[2]	[3]	[4]	[5]	[6]	[7]

- Example 3 x 4 array:

a b c d
e f g h
i j k l

- Convert into 1D array y by collecting elements by columns.
- We get y = {a, e, i, b, f, j, c, g, k, d, h, l}

- Example 3 x 4 array:

a b c d
e f g h
i j k l

- Convert into 1D array y by collecting elements by rows.
- We get y[] = {a, b, c, d, e, f, g, h, i, j, k, l}

Χρήση Πινάκων (I)

- Στη CLANG οι πίνακες πρέπει να ορίζονται ως global μεταβλητές.
- Όλοι οι πίνακες στην CLANG καλό είναι να χρησιμοποιείτε 1D arrays.
- Η πρόσβαση στους πίνακες της CLANG θα γίνεται με χρήση αριθμητικής διευθύνσεων.
- Παραδείγματα:

C	CLANG
<pre>int x[3] = {1, 2, 3}; x[2] = x[1] + 3;</pre>	<pre>int x[3] = {1, 2, 3}; R1 = 1; R2 = x[R1]; R3 = R2 + 3; R2 = 2; x[R2] = R3;</pre>

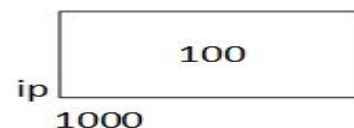
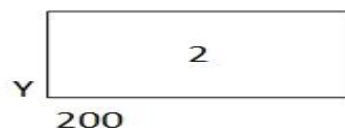
Χρήση Πινάκων (II)

C	CLANG
<pre>int x[2][3] = {{1, 2, 3}, {4,5,6}}; x[1][2] = x[1][1] + 5;</pre>	<pre>int x[6] = {1, 2, 3, 4, 5, 6}; R1 = 1; R2 = 2; R3 = R1*3 + R1; R4 = x[R3]; R5 = R4 + 5; R3 = R1*3 + R2; X[R3] = R5;</pre>

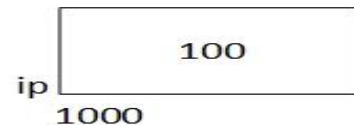
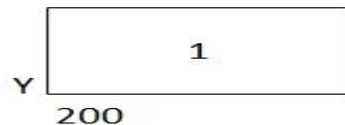
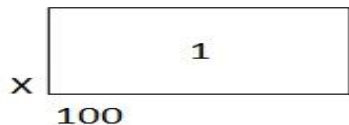
Εισαγωγικά για δείκτες

- Κάθε μεταβλητή σχετίζεται με μια θέση στην κύρια μνήμη του Η/Υ η οποία έχει τη δική της ξεχωριστή διεύθυνση.
- Ο δείκτης είναι μια μεταβλητή η οποία χρησιμοποιείται για την αποθήκευση μιας διεύθυνσης της κύριας μνήμης του Η/Υ

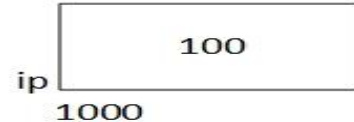
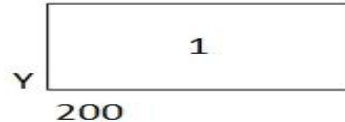
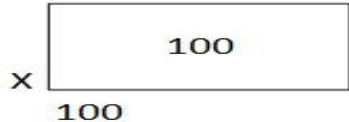
```
int X = 1, Y = 2;  
int *ip;  
ip = &X;
```



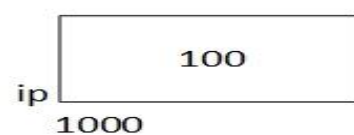
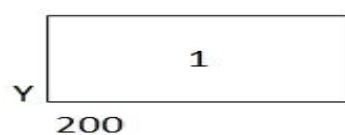
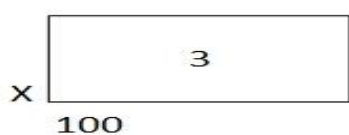
```
Y = *ip;
```



```
X = ip;
```



```
*ip = 3;
```



Το type casting στη C.

- Η μέθοδος του type casting στη C χρησιμοποιείται για να αλλάξουμε τύπο σε μια μεταβλητή.
- Για το εργαστήριο θα χρειαστούμε μετατροπή τύπου από int σε structure pointer και αντίστροφα.
- Το type casting μπορεί να γίνει οποιαδήποτε στιγμή στον κώδικα και αναφέρεται σε κάθε μεταβλητή που χρησιμοποιούμε.

Pointers σε CLANG

- Στη CLANG χρησιμοποιούμε για όλες τις μεταβλητές μας χρησιμοποιούμε καταχωρητές.
- Οι καταχωρητές είναι τύπου int. Για να διαχειριστούμε pointers θα πρέπει να κάνουμε type casting.

Παραδείγμα 1: Δημιουργίας λίστας

C	CLANG
<pre>struct list* p = malloc(sizeof(struct list)); p->id = 1; p->value = 1; p->next = NULL;</pre>	<pre>struct list* p; R24 = (int)(malloc(sizeof(struct list))); p = (struct list *)R24; R25 = 1; //R25 is id R26 = 1; //R26 is value R27 = NULL; p->id = R25; p->value = R26; p->next = R27; R24 = (int) p; //R24 is head</pre>

Pointers σε CLANG (συν.)

- Στην Assembly υπάρχουν συγκεκριμένοι καταχωρητές που περνάνε τα ορίσματα στις συναρτήσεις αυτοί είναι οι R4 έως R7.
- Επίσης είναι συγκεκριμένος και ο καταχωρητής που περιέχει την τιμή επιστροφής της συνάρτησης, αυτός είναι ο R2.
- Άρα ανεξάρτητα το όρισμα της συνάρτησης πρέπει να γίνεται type casting σε int και να μπαίνει σε έναν από τους καταχωρητές ορισμάτων.
- Πλέον όλες οι συναρτήσεις επιστρέφουν integer!

I/O functions

- Οι συναρτήσεις που θα χρησιμοποιείτε θα είναι ίδιες με αυτές της C.

C	CLANG
<pre>int x = 5; printf("%d\n", x);</pre>	<pre>int x = 5; R1 = x; printf("%d\n", x);</pre>
<pre>int x; scanf("%d\n", &x)</pre>	<pre>int x; scanf("%d\n", &R1) x = R1;</pre>

Σ Παράδειγμα 1: Add two Arrays

```
int a[4] = {1, 2, 3, 4};
```

```
int b[4] = {15, 16, 17, 18};
```

```
int sum[4];
```

```
int main()
```

```
{
```

```
    int i = 0;
```

```
    for (i = 0; i < 4; i++)
```

```
    {
```

```
        sum[i] = a[i] + b[i];
```

```
    }
```

```
    for (i = 0; i < 4; i++)
```

```
    {
```

```
        printf("%d\n",sum[i]);
```

```
    }
```

```
    return(0);
```

```
}
```

CLANG Παράδειγμα 1: Add two Arrays (I)

```
int a[4] = {1, 2, 3, 4};  
int b[4] = {15, 16, 17, 18};  
int sum[4];
```

```
int R0 = 0;
```

```
int R1, R2, R3, R4, R5, R6, R7, R8, R9, R10, R11, R12, R13, R14, R15, R16, R17, R18, R19, R20, R21, R22, R23, R24, R25,  
    R26, R27, R28, R29, R30, R31;
```

```
void main()  
{
```

```
    int i = 0;
```

```
    R1 = i;
```

```
    R1 = 0;
```

```
    R2 = 4;
```

```
while_label: if(R1 >= R2) goto end_loop;
```

```
    R3 = a[R1];
```

```
    R4 = b[R1];
```

```
    R5 = R3 + R4;
```

```
    sum[R1] = R5;
```

```
    R1 = R1 + 1;
```

```
    goto while_label;
```

CLANG Παράδειγμα 1: Add two Arrays (II)

```
end_loop:
```

```
    R1 = 0;
```

```
while_label2: if(R1 >= R2) goto end_loop2;
```

```
    R3 = sum[R1];
```

```
    printf("%d\n",R3);
```

```
    R1 = R1 + 1;
```

```
    goto while_label2;
```

```
end_loop2:
```

```
    i = 5;
```

```
}
```

C Παράδειγμα 2: Add two Matrices

```
int a[3][4] = {{1,2,3,4},{4,5,6,7},{7,8,9,10}};  
int b[3][4] = {{33,34,35,35},{36,37,38,39},{39,40,41,42}};  
int result[3][4];
```

```
void main()  
{  
    int i = 0, j = 0;  
  
    for(i = 0; i < 3; i++)  
    {  
        for(j = 0; j < 4; j++)  
        {  
            result[i][j] = a[i][j] + b[i][j];  
        }  
    }  
  
    for(i = 0; i < 3; i++)  
    {  
        for(j = 0; j < 4; j++)  
        {  
            printf("%d\n", result[i][j]);  
        }  
    }  
}
```

CLANG Παράδειγμα 2: Add two Matrices

```
#include <stdio.h>
```

```
int a[12] = {1,2,3,4,4,5,6,7,7,8,9,10};
```

```
int b[12] = {33,34,35,35,36,37,38,39,39,40,41,42};
```

```
int result[12];
```

```
int R0 = 0;
```

```
int R1, R2, R3, R4, R5, R6, R7, R8, R9, R10, R11, R12, R13, R14, R15, R16, R17, R18, R19, R20, R21, R22, R23, R24, R25,  
    R26, R27, R28, R29, R30, R31;
```

```
void main()
```

```
{
```

```
    int i = 0, j = 0;
```

```
    R1 = i;
```

```
    R2 = j;
```

```
    R3 = 3;
```

```
    R4 = 4;
```

```
    for_loop1:
```

```
        if ( R1 >= R3) goto end_loop1;
```

```
        R2 = 0;
```

```
        for_loop2: if( R2 >= R4) goto end_loop2;
```

CLANG Παράδειγμα 2: Add two Matrices

```
R5 = R1*R4 + R2;
```

```
R6 = a[R5];
```

```
R7 = b[R5];
```

```
R8 = R6 + R7;
```

```
result[R5] = R8;
```

```
R2 = R2 + 1;
```

```
goto for_loop2;
```

```
end_loop2:
```

```
R1 = R1 + 1;
```

```
goto for_loop1;
```

```
end_loop1:
```


CLANG Παράδειγμα 2: Add two Matrices

```
R1 = 0;
  for_loop3:
    if ( R1 >= R3) goto end_loop3;

    R2 = 0;

    for_loop4: if( R2 >= R4) goto end_loop4;

        R5 = R1*R4 + R2;
        R6 = result[R5];
        printf("%d\n",R6);

        R2 = R2 + 1;
        goto for_loop4;

    end_loop4:

    R1 = R1 + 1;
    goto for_loop3;

  end_loop3:
}
```

Σ Παράδειγμα 3: Condition AND

```
int x = 5, y = 8, z;
```

```
if(x <= 7 && y > 10)
```

```
    z = x + y;
```

```
else
```

```
    z = x - y;
```

CLANG Παράδειγμα 3: Condition AND

```
int x = 5, y = 8, z;  
R1 = x;  
R2 = y;  
    if ( R1 > 7) goto else_label;  
    If (R2 <= 10) goto else_label;  
        R3 = R1 + R2;  
        goto end_if;  
else_label:  
        R3 = R1 - R2;  
End_if:  
    z = R3;
```

Σ Παράδειγμα 4: Αναδρομή

```
#include<stdio.h>
```

```
int Fibonacci(int n)
```

```
{
```

```
    if ( n == 0 )
```

```
        return 0;
```

```
    else if ( n == 1 )
```

```
        return 1;
```

```
    else
```

```
        return ( Fibonacci(n-1) + Fibonacci(n-2) );
```

```
}
```

CLANG Παράδειγμα 4: Αναδρομή

```
int Fibonacci(int R2)
{
    int temp1;

    if(R2 != 0) goto lab_else_if;
        R9 = 0;
        goto lab_end;
lab_else_if:
    if (R2 != 1 ) goto lab_else;
        R9 = 1;
        goto lab_end;
lab_else:

    R5 = R2-1;
    R7 = Fibonacci(R5);
    temp1 = R7;

    R6 = R2-2;
    R8 = Fibonacci(R6);
    R7 = temp1;
    R9 = R7+R8;

lab_end:
    return(R9);
}
```