ΤΠ 2104 Αρχιτεκτονική Υπολογιστών (Εργαστήριο)

ΑΝΩΤΑΤΟ ΤΕΧΝΟΛΟΓΙΚΟ

ΕΚΠΑΙΔΕΥΤΙΚΟ ΙΔΡΥΜΑ ΚΡΗΤΗΣ

ΤΕΧΝΟΛΟΓΙΚΟ ΕΚΠΑΙΔΕΥΤΙΚΟ ΙΔΡΥΜΑ ΚΡΗΤΗΣ

Σειρά Ασκήσεων 6:

Κάλεσμα Διαδικασιών στον MIPS

Το κάλεσμα διαδικασίας (procedure call) γίνεται μέσω της εντολής `jal` (`jump and link` - άλμα και σύνδεση), η δε επιστροφή από διαδικασία (procedure return) γίνεται μέσω της εντολής `jr` (`jump register` -άλμα εκεί που δείχνει ένας καταχωρητής), όπως είδαμε στην § 5.1.

Τίνος Ευθύνη είναι το Σώσιμο των Καταχωρητών

Οταν διάφορες διαδικασίες χρησιμοποιούν τον ίδιο καταχωρητή γιά διαφορετικούς σκοπούς η καθεμία, σε τρόπο που να καταστρέφεται η προηγούμενη τιμή του καταχωρητή --την οποία όμως δεν θέλουμε να χάσουμε, τότε καποιος πρέπει να "σώσει" (αντιγράψει) την προηγούμενη τιμή σε ασφαλές μέρος (στη μνήμη) και αργότερα να την επαναφέρει στον καταχωρητή. Γιά να ελαχιστοποιηθεί το κόστος αυτής της εργασίας (δηλαδή το πόσο συχνά πρέπει αυτή να γίνεται), οι compilers του MIPS ακολουθούν τις παρακάτω συμβάσεις. (Οι συμβάσεις αυτές είναι αρκούντως συντηρητικές ώστε να επιτρέπουν χωριστή μετάφραση (separate compilation) των διαδικασιών). Εστω ότι μιά διαδικασία A καλεί μιάν άλλη διαδικασία B (ή η ενεργοποίηση (instance) A μιάς αναδρομικής διαδικασίας καλεί την ενεργοποίηση B του εαυτού της). Θα ονομάζουμε την:

- A: "η **καλούσα** διαδικασία" (**caller**), και την
- B: "η **καλουμένη** διαδικασία" (**callee**).

Προσωρινοί Καταχωρητές $t0 - $t9 (`$8-$15` και `$24-$25`) -- temporary registers: Η τιμή των καταχωρητών αυτών δεν διατηρείται μετά από ένα κάλεσμα διαδικασίας (not preserved across call), δηλαδή η καλούμενη διαδικασία (ή άλλες που τυχόν καλιούνται από αυτήν) επιτρέπεται να μεταβάλει την τιμή αυτών των καταχωρητών χωρίς προηγουμένως να σώσει την τιμή που τυχόν είχε μείνει εκεί από την καλούσα διαδικασία, επομένως και χωρίς να επαναφέρει την παλαιά εκείνη τιμή προ της επιστροφής στην καλούσα διαδικασία. Αρα, αν η καλούσα διαδικασία έχει κάτι χρήσιμο μέσα σε έναν τέτοιο καταχωρητή ενώ ετοιμάζεται να καλέσει μιάν άλλη διαδικασία, το οποίο χρήσιμο επιθυμεί να το ξαναβρεί στη θέση του μετά την επιστροφή του καλέσματος, είναι **ευθύνη της καλούσας διαδικασίας** να σώσει την χρήσιμη τιμή πριν το κάλεσμα ("**caller-saved**"), και να την επαναφέρει αμέσως μετά από αυτό (δηλ. μετά την επιστροφή του). Προσωρινές τιμές των οποίων η διάρκεια ζωής (lifetime) δεν περιλαμβάνει καλέσματα διαδικασιών συμφέρει να τοποθετούνται σε τέτοιους καταχωρητές, διότι δεν χρειάζεται να σώσουμε τα παλαιά περιεχόμενα (από αυτόν που μας κάλεσε) αυτών των καταχωρητών πριν τους χρησιμοποιήσουμε.

[*Σημείωση:* "διάρκεια ζωής (lifetime)" μιάς μεταβλητής ή ενός καταχωρητή λέγεται η χρονική περίοδος (ή οι εντολές που περιλαμβάνονται) από τη στιγμή πού στη μεταβλητή ή στον καταχωρητή εκχωρείται μία τιμή, μέχρι την τελευταία ανάγνωση της τιμής αυτής πριν την επόμενη εκχώρηση νέας τιμής σε αυτή τη μεταβλητή ή τον καταχωρητή. Μετά το τέλος μιάς διάρκειας ζωής μιάς μεταβλητής ή ενός καταχωρητή, και μέχρι την έναρξη της επόμενης διάρκειας ζωής της/του, η μεταβλητή ή ο καταχωρητής θεωρείται "νεκρή/νεκρός", διότι η τιμή της/του **δεν** μας ενδιαφέρει --κανείς δεν πρόκειται να την διαβάσει πριν την ξαναλλάξουμε-- άρα είμαστε ελεύθεροι να την καταστρέψουμε].

Διατηρούμενοι Καταχωρητές $s0 - $s7 (`$16-$23`) -- saved registers: Η τιμή των καταχωρητών αυτών διατηρείται μετά από ένα κάλεσμα διαδικασίας (preserved across call), δηλαδή είναι **ευθύνη της καλούμενης διαδικασίας** να σώσει την παλαιά τιμή κάθε τέτοιου καταχωρητή πριν την χαλάσει ("**callee-saved**"), και να την επαναφέρει στη θέση της πριν επιστρέψει στην καλούσα διαδικασία. Αρα, η καλούσα διαδικασία μπορεί να αφήνει χρήσιμες τιμές σε αυτούς τους καταχωρητές, πριν καλέσει άλλες διαδικασίες, και να τις ξαναβρίσκει μετά την επιστροφή από αυτές, χωρίς να χρειάζεται --η καλούσα διαδικασία-- να κάνει κάτι ιδιαίτερο γι' αυτό. Μεταβλητές και τιμές των οποίων η διάρκεια ζωής (lifetime) περιλαμβάνει πολλά καλέσματα διαδικασιών, με επανειλημμένη χρήση της παλαιάς τιμής τους μεταξύ των καλεσμάτων, συμφέρει να τοποθετούνται σε τέτοιους καταχωρητές, διότι, αν οι καλούμενες διαδικασίες (και οι τυχόν δικοί τους απόγονοι) δεν χρησιμοποιούν αυτούς τους καταχωρητές, γλιτώνουμε τα πολλαπλά σωσίματα και επαναφορές (σώζουμε και επαναφέρουμε μόνο μία φορά την τιμή που είχε αφήσει εκεί η διαδικασία που

κάλεσε εμάς τους ίδιους).

Άλλοι Καταχωρητές

$ra ($31) Return Address: κάθε κάλεσμα διαδικασίας καταστρέφει το προηγούμενο περιεχόμενο του καταχωρητή αυτού, άρα κάθε διαδικασία που καλεί άλλες διαδικασίες (δηλ. κάθε διαδικασία που δεν είναι φύλλο στο δέντρο των καλεσμάτων) πρέπει να σώσει τη διεύθυνση επιστροφής της στην αρχή της εκτέλεσής της (πριν το πρώτο κάλεσμα), και να την επαναφέρει αφού επιστρέψει το τελευταίο παιδί της, πριν να επιστρέψει η ίδια. (Το σημείο αυτό είναι ασαφές στο σχήμα 3.11 του βιβλίου (σελ. 138): η διεύθυνση επιστροφής στην οποία αναφέρεται εκείνο το σχήμα είναι η διεύθυνση του παρόντος καλέσματος, και όχι η διεύθυνση επιστροφής της διαδικασίας μέσα από την οποία γίνεται το κάλεσμα).

$sp ($29) Stack Pointer, $fp ($30) Frame Pointer: όποιος τους αλλάζει τους επαναφέρει κιόλας. Πολλοί compilers δεν χρησιμοποιούν frame pointer --το ίδιο κάνει και το βιβλίο στην παράγρ. 3.6, και το ίδιο θα κάνουμε και εμείς. Γιά το πώς θα μπορούσε να χρησιμοποιηθεί ο frame pointer δείτε (προαιρετικά) την παράγρ. Α.6 του Παραρτήματος του βιβλίου.

$gp ($28) Global Area Pointer: Δεν τον αλλάζει (ούτε τον σώζει) καμία διαδικασία. Δείχνει στη μέση μιάς περιοχής μνήμης μεγέθους 64 KBytes τα περιεχόμενα της οποίας μπορούμε να προσπελάσουμε με μία μόνο εντολή load ή store χρησιμοποιώντας αυτόν τον καταχωρητή, και στην οποία επομένως συμφέρει να τοποθετούνται οι γενικές (global) βαθμωτές (scalar) μεταβλητές του προγράμματος.

$k0-$k1 ($26-$27) Kernel Reserved, $at ($1) Assembler Temporary: δεν τους χρησιμοποιούν οι compilers και τα προγράμματα χρήστη --κρατημένοι γιά το λειτουργικό σύστημα και τον Assembler.

$zero ($0) = 0: περιέχει την σταθερά (hardwired) μηδέν. Επιτρέπονται εγγραφές σε αυτόν, αλλά **δεν** αλλάζουν το μηδενικό (hardwired) περιεχόμενο του.

$a0-$a3 ($4-$7) Procedure Argument Registers: περιέχουν τα πρώτα 4 ορίσματα (arguments) της διαδικασίας. Αν υπάρχουν περισσότερα από 4 ορίσματα, τα υπόλοιπα περνιούνται στη στοίβα. Αν υπάρχουν λιγότερα από 4 ορίσματα, τότε οι υπόλοιποι (μη χρησιμοποιούμενοι) καταχωρητές `$a3-$a0` χρησιμοποιούνται σαν τους προσωρινούς καταχωρητές `$t0-$t9`. Π.χ., αν η παρούσα διαδικασία έχει δύο ορίσματα μόνο, τότε είναι ελεύθερη να χρησιμοποιήσει τους `$a2, $a3` σαν προσωρινούς καταχωρητές χωρίς να σώσει τις τιμές τους, αρκεί να μην ζητά να διατηρούνται αυτές οι τιμές όταν καλεί παιδιά της: αν τα παιδιά της έχουν πολλά ορίσματα τότε πρέπει να τους χρησιμοποιήσει γιά να βάλει εκεί τα ορίσματά τους, κι αν έχουν λίγα ορίσματα τότε επιτρέπεται τα παιδιά αυτά να χαλάσουν τις τιμές των `$a2, $a3`. Κατ' αναλογία, και οι `$a0, $a1` περιέχουν τα ορίσματα της παρούσας διαδικασίας μόνο μέχρι το κάλεσμα του πρώτου παιδιού της --γιά το κάλεσμα αυτό, οι καταχωρητές αυτοί πρέπει να χρησιμοποιηθούν γιά τα ορίσματα των παιδιών (ή σαν προσωρινοί των παιδιών αν τα ορίσματά τους είναι λιγότερα).

$v0-$v1 ($2-$3) Procedure Return Values: περιέχει (ο `$v0`) την τιμή που κάθε διαδικασία επιστρέφει στην καλούσα διαδικασία (ανάλογα και ο `$v1` γιά την περίπτωση επιστροφής δύο τιμών). Η τιμή που επεστράφη ισχύει μόνο μέχρι το επόμενο κάλεσμα διαδικασίας. Ο μη χρησιμοποιούμενος καταχωρητής επιστροφής

Πού Σώζονται οι Καταχωρητές

Αν οι διαδικασίες δεν ήταν αναδρομικές, δηλαδή αν απαγορεύονταν να καλέσει μιά διαδικασία τον εαυτό της --είτε άμεσα είτε έμμεσα μέσω άλλων που αυτή καλεί, τότε θα αρκούσε μιά συγκεκριμένη περιοχή στη μνήμη γιά κάθε διαδικασία, όπου αυτή να φυλάει τις τιμές των καταχωρητών που πρέπει να σώσει και αργότερα να επαναφέρει. Ομως, οι σημερινές γλώσσες προγραμματισμού επιτρέπουν την αναδρομή (επανενεργοποιούμενες διαδικασίες, re-entrant procedures), κι έτσι μιά τέτοια λύση δεν θα δούλευε. Δεδομένου ότι τα κλέσματα και οι επιστροφές διαδικασιών λειτουργούν με τρόπο "last in first out" (LIFO), η φυσική δομή δεδομένων γιά δυναμική παραχώρηση και απελευθέρωση μνήμης στις διαδικασίες και από τις διαδικασίες είναι η **στοίβα** (stack).

Η στοίβα (χρήστη) στον MIPS ξεκινάει από τη διεύθυνση 7f.ff.ff.ff (δεκαεξαδικό), δηλαδή από τη μέση της μνήμης, και μεγαλώνει προς τις μικρότερες διευθύνσεις (προς τη διεύθυνση 0). Ο `$sp` δείχνει στην τελευταία χρησιμοποιούμενη λέξη της στοίβας. Πριν αποθηκεύσουμε N νέες λέξεις στη στοίβα πρέπει να ελαττώσουμε

τον `$sp` κατά 4N, πράγμα που ισοδυναμεί με την δήλωση από πλευράς του προγράμματός μας (προς το λειτουργικό σύστημα, σε περίπτωση διακοπής (interrupt)) ότι τώρα η στοίβα μας είναι μεγλύτερη κατά N λέξεις. Η αντίστροφη πράξη (αύξηση του `$sp` κατά 4N --απελευθέρωση μνήμης) πρέπει να γίνει αφού πάρουμε τις αποθηκευμένες λέξεις και δεν τις χρειαζόμαστε άλλο πιά στη στοίβα. Οι τιμές που βρίσκονται αποθηκευμένες στη στοίβα προσπελαύνονται συνήθως μέσω εντολών load και store με διευθυνσιοδότηση σχετικά με τον `$sp`. Καθώς ο `$sp` αυξομειώνεται λόγω παραχώρησης/απελευθέρωσης μνήμης, η απόσταση των αποθηκευμένων τιμών στη στοίβα από τον `$sp` αλλάζει, αλλά παραμένει πάντοτε γνωστή στον compiler/προγραμματιστή (εκτός των περιπτώσεων δυναμικής παραχώρησης μνήμης στη στοίβα --πράγμα σπάνιο στις γλώσσες προγραμματισμού-- οπότε και απαιτείται η χρήση του `$fp`).

Καθολικές (Global) και Τοπικές (Local) Μεταβλητές

Οι "καθολικές" (global) και οι "τοπικές" (local) μεταβλητές είναι έννοιες των γλωσσών προγραμματισμού υψηλού επιπέδου (HLL - π.χ. C, κλπ). Γιά τον Assembler δεν υπάρχουν αυτές οι έννοιες, ενώ η υλοποίηση των εννοιών αυτών σε επίπεδο γλώσσας Assembly επαφίεται στον προγραμματιστή (ή στον compiler). Στις HLL, μιά καθολική μεταβλητή αντιστοιχεί πάντα σε μία δεδομένη, σταθερή θέση μνήμης (ή καταχωρητή), ανεξαρτήτως του ποιά διαδικασία εκτελείται κάθε στιγμή· η τιμή μιάς καθολικής μεταβλητής ούτε χρειάζεται να αποθηκευτεί ποτέ στη στοίβα, ούτε επανατίθεται ποτέ σε παλαιές τιμές της από τη στοίβα. Αντίθετα, μιά τοπική μεταβλητή έχει νόημα μόνο όσο είναι ενεργή η διαδικασία στην οποία ανήκει, και είναι ανύπαρκτη πριν την εκκίνηση της διαδικασίας αυτής ή μετά τον τερματισμό (επιστροφή) της διαδικασίας· εάν η διαδικασία επιστρέψει και ξανακαλεστεί, η παλαιά τιμή της τοπικής μεταβλητής έχει χαθεί. Επίσης, τοπικές μεταβλητές με το ίδιο όνομα αλλά σε διαφορετική διαδικασία (ή σε διαφορετική ενεργοποίηση της ίδιας (αναδρομικής) διαδικασίας!) είναι διαφορετικές και άσχετες μεταξύ τους!

Συνήθως, οι μεταφραστές (compilers) τοποθετούν την κάθε καθολική μεταβλητή σε μία σταθερή διεύθυνση μνήμης --όχι στη στοίβα· η διεύθυνση αυτή δεν αλλάζει από διαδικασία σε διαδικασία. Στην παρούσα άσκηση, γιά λόγους απλότητας, θα κρατήσετε τις καθολικές μεταβλητές σε καταχωρητές· όλες οι διαδικασίες θα χρησιμοποιούν τις τιμές που θα βρίσκουν σε αυτούς τους καταχωρητές, και θα αφήνουν εκεί τις νέες τιμές που θα θέλουν να αφήσουν προκειμένου να τις βρούν στη συνέχεια όποιες άλλες διαδικασίες κληθούν ή επανενεργοποιηθούν μετά την επιστροφή παιδιών τους. Καμία διαδικασία δεν επαναφέρει στις καθολικές αυτές μεταβλητές (δηλαδή στους καταχωρητές αυτούς) παλαιές τιμές τους που είχαν αποθηκευτεί κάπου (άρα και κανείς δεν χρειάζεται να αποθηκεύει τιμές τους πουθενά). Επίσης, καμιά διαδικασία δεν μπορεί να χρησιμοποιήσει έναν τέτοιο καταχωρητή γιά άλλη δουλεία από την καθολική μεταβλητή που αυτός κρατά.

Αντίθετα, οι τοπικές μεταβλητές αντιστοιχούν σε μιά θέση στη στοίβα, σε δεδομένη απόσταση **σχετικά** με τον stack-pointer, η οποία θέση υπάρχει μόνον όση ώρα είναι ενεργοποιημένη η αντίστοιχη διαδικασία, και η οποία θέση --σαν απόλυτη διεύθυνση μνήμης-- πολύ πιθανόν να αλλάζει από ενεργοποίηση σε ενεργοποίηση της διαδικασίας. Εάν ένα αντίτυπο της μεταβλητής κρατηθεί σε καταχωρητή (ή, σαν πολύ συνηθισμένη βελτιστοποίηση, κρατηθεί μόνο το "αντίτυπο", χωρίς το πρωτότυπο), τότε η διαδικασία αυτή (σε συνεργασία με τις άλλες) έχει την ευθύνη να σώζει το αντίτυπο στη στοίβα και να το επαναφέρει όποτε υπάρχει κίνδυνος μιά άλλη διαδικασία να χρησιμοποιήσει τον καταχωρητή αυτό διαφορετικά, όπως αναλύσαμε παραπάνω.

Παρατηρήστε ότι οι παραπάνω έννοιες δεν είναι ίδιες με τις "καθολικές ετικέτες" (global labels) που γιά τον Assembler ορίζονται με την οδηγία ".globl". Οι ετικέτες του Assembler είναι απλά διευθύνσεις μνήμης --είτε δεδομένων, είτε εντολών μέσα σε πρόγραμμα, είτε οτιδήποτε. Καθολική ετικέτα είναι απλώς μιά διεύθυνση την οποία ζητάμε από τον Assembler να την θυμάται, μαζί με το συμβολικό της όνομα, και μετά τη λήξη της "μετάφρασης" από Assembly σε γλώσσα μηχανής του κομματιού προγράμματος όπου την βρήκε, ώστε να μπορούμε να αναφερθούμε σε αυτήν και στον μέλλον –πιθανά μέσα από άλλα αρχεία με άλλα κομάτια προγράμματος Assembly.

Άσκηση 5.1: Συνάρτηση-φύλλο

Δίνεται ο παρακάτω κώδικας σε C. Μεταφράστε τον κώδικα σε Assembly του MIPS. Οι μεταβλητές g, h, i, j αντιστοιχίζονται στους καταχωρητές $a0, $a1, $a2, $a3 και η f στον $s0. Τυπώστε μία ενδιαφέρουσα χρονική στιγμή τρεξίματος, όπου να φαίνονται τα δεδομένα της στοίβας και οι τιμές των καταχωρητών του MIPS.


```
int leaf_example(int g,int h, int i, int j)
{
        int f;
        f = (g+h) – (i+j);
        return f;
}
```

Η main θα μπορουσε να ειναι ως εξής:

```
main()
{
int result;
result = leaf_example(0, 1,2, 3);
}
```