Examen de compilation

M1 Informatique Fondamentale, ENS Lyon
Jeudi 10 janvier 2013

Durée: 3 heures. Tous les documents sont autorisés.
Cet examen est constitué de 4 exercices indépendants.

Exercice 1. *Traduction dirigée par la syntaxe*

On considère le programme suivant:

```
typedef struct {
  int im;
  int re;
} complex_t;

void foo()
{
  complex_t[2] tab;
  int i;

//point 1
  for(i=0; i<2; i++)
    {
      tab[i].im = i;
      tab[i].re = i+1;
    }
}

void main()
{
  foo();
}
```

Q1) Sans détailler, donner l'état de la pile après l'exécution du prélude de foo (au point 1).
On supposera que la fonction main() n'a pas d'enregistrement d'activation, et que l'enregistrement d'activation de foo() commence à l'octet 255.

Q2) En détaillant, donner $[\![\text{complex\_t[2]}]\!]^{\texttt{temp}}$.
On laissera tels quels les temporaires current_cell, counter, etc, des règles de traduction, sans chercher à les remplacer par leur valeur. De même, on écrira telles quelles les constructions "macro" comme SP = SP - 2.

Q3) En détaillant, donner $[\![\text{tab[1].re} = 2]\!]_\rho$.
On supposera que index(complex_t,re) = 1. Sur le slide de cours avec les règles de traduction utiles pour cette question, il faut lire "- index", au lieu de "+ index".

T.S.V.P.

Exercice 2. *Forme SSA*

La compilation d'une boucle `for` produit le code intermédiaire suivant:

```
  r9 = 0
for_0:
  r22 = 10
  cjump r9 < r22 --> lower_0
  r23 = 0
  jump not_lower_0
lower_0:
  r23 = 1
not_lower_0:
  r24 = 1
  cjump r23 == r24 --> continue_for_0
  jump end_for_0
continue_for_0:
  r9 = r9 + 1
  jump for_0
end_for_0:
```

Q4) Sans détailler, tracer le CFG et l'arbre de dominance.

Q5) Sans détailler, donner les ensembles DF_{local} et DF_{up} pour chaque bloc.

Q6) On rappelle que la frontière de dominance d'un bloc B est donnée par:

$$DF(B) = DF_{\text{local}}(B) \cup \bigcup_{B \text{ idom } C} DF_{\text{up}}(C)$$

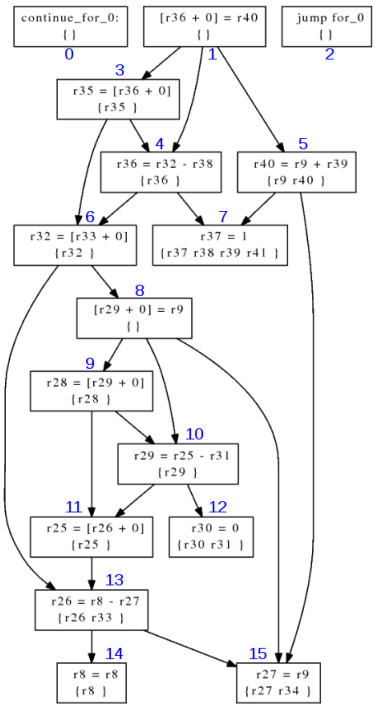
Sans détailler, donner la frontière de dominance pour chaque bloc. Pourquoi l'union se borne-t-elle à parcourir les blocs C immédiatement dominés?

Q7) En détaillant, passer le programme sous forme SSA.

Q8) En détaillant, sortez le programme de la forme SSA.

Exercice 3. *Ordonnancement*

La compilation du programme donné en exercice 1 produit le DAG suivant pour le corps de la boucle:



Q9) Pourquoi y a-t-il un l'arc de 1 vers 3?

Q10) On souhaite appliquer l'heuristique naïve vue en cours. Sans détailler, donner une décomposition du DAG en sous-arbres.

Q11) Faut-il produire du code pour les feuilles qui se trouvent être des noeuds partagés? Comment les gérer?

Q12) En justifiant, indiquer dans quel ordre les arbres doivent être évalués.

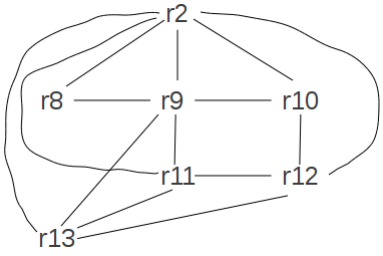
Q13) On considère l'arbre enraciné en 6. En détaillant, produire le code en appliquant l'algorithme de Sethi-Ullman.

Q14) Sans détailler, tracer les durées de vies sur le code obtenu. Donner la pression registre. Est-elle conforme à celle calculée en 13)? Pourquoi?

Q15) Ici, le temporaire r2 est live-out. Quelle précaution faut-il prendre lors de la génération du code?

Exercice 4. *Allocation de registres*

Après avoir ordonnancé le code machine, on obtient le graphe d'interférence suivant. On suppose que le noeud r2 est précoloré avec le registre r2.



Q16) Que représentent les noeuds? Pourquoi précolorer le noeud r2?

Q17) Trouver une allocation des registres en appliquant l'algorithme de Chaitin pour $K = 3$ registres, détailler.

Q18) Comment/où allouer les variables spillées?