Baumstrukturierte Adressen

Die Regeln rund um die Befehle `JMP` und `JMC` sind für C_0 und C_1 gleich, deswegen mache ich das hier mal für C_0 und die Übersetzung in AM_0. Außerdem ist das Verfahren für alle Konstrukte mit Sprüngen (also if-then, if-then-else und while Statements) ähnlich. Ich glaube, es ist ausreichend, wenn ich es hier beispielhaft an einem if-then-else-Statement mache, für den Rest kannst du für Details einfach nochmal ins Skript schauen, das ist dann wirklich analog. Also los geht's ...

Sehen wir uns einmal die Übersetzung aus dem Skript an. Da gibt es ja die Übersetzungsfunktionen *trans*, *blocktrans*, *stseqtrans*, aber das wichtige für uns ist jetzt die Funktion *sttrans*. Da gibt es nun mehrere Definitionen, aber wir wollen ein if-then-else-Statement übersetzen, d.h. nur eine Definition ist wichtig für uns, nämlich die Folgende:

$$
\begin{aligned}
sttrans(\texttt{if (} exp \texttt{)} \ stat_1 \ \texttt{else} \ stat_2, tab, a) := \ & boolexptrans(exp, tab) \\
& \texttt{JMC } a.1 \\
& sttrans(stat_1, tab, a.2) \\
& \texttt{JMP } a.3 \\
a.1: \ & sttrans(stat_2, tab, a.4) \\
a.3: \ &
\end{aligned}
$$

Da wird nun das C_0-Statement in die Funktion gegeben, zusätzlich noch eine Symboltabelle *tab* (nicht so wichtig jetzt, da stehen nur die Hauptspeicheradressen drin), und eine Basisadresse a. Die Basisadresse a kommt dabei aus den vorherigen Übersetzungsfunktionen. Meist ist dies *stseqtrans*, welches jedem Statement der Reihe nach eine Adresse zuweist. So bekommt dann das erste Statement einer Sequenz die Basisadresse 1, das zweite die Adresse 2 usw. Haben wir aber verschachtelte Konstrukte (z.B. ein if-then-else in einer while-Schleife), dann kann die Basisadresse, mit der wir in die Übersetzung rein gehen, auch anders sein. Insbesondere bei C_1 kommt noch die Nummer der Funktion davor.
In deinem Beispiel (Aufgabe 10.1) haben wir erst zwei `scanf`-Befehle, die bekommen die Basisadressen 1.1 bzw. 1.2. Die 1 am Anfang kommt dabei aus der Tatsache, dass wir uns in der Main-Funktion befinden (das ist bei C_0 ein wenig trivial, aber schau dir einfach mal die Definition von *blocktrans* an, da setzt man halt einfach die 1 vorn ran). Das relevante if-Statement kommt an dritter Stelle und bekommt somit die Basisadresse 1.3, also $a = 1.3$.

Nun übersetzt man die Bedingung *exp*, einfach so wie wir es immer gemacht haben. Anschließend kommt der erste Sprungbefehl `JMC`. Laut Definition bekommt der die Sprungmarke $a.1$ und wegen $a = 1.3$ kommt also dort 1.3.1 hin.

Anschließend wird der then-Teil übersetzt. Dieser bekommt intern die Basisadresse $a.2 = 1.3.2$. Meist kommt diese Adresse nie zum Vorschein, weil wir sie nicht benötigen, sofern wir keine Konstrukte mit Sprüngen in $stat_1$ übersetzen müssen. Steht aber z.B. eine while-Schleife im then-Zweig, dann bekommt sie die Basisadresse 1.3.2. In den meisten Fällen haben wir nur simple Berechnungen wie `x = x + 1` und da benötigen wir keine solche Adresse, sodass sie nie erscheinen wird.

Haben wir den then-Zweig fertig übersetzt, müssen wir die Ausführung des else-Zweiges mit einem Sprung verhindern. Man könnte nun denken, da wir $a.1$ verwendet haben kommt als nächstes $a.2$, aber das passiert nicht, weil $a.2$ ist ja schon an den then-Zweig weitergereicht und dementsprechend nicht mehr verfügbar. Also nehmen wir die nächste freie Adresse, und das ist $a.3$ bzw. in unserem Beispiel 1.3.3.

Schließlich wird noch der else-Teil übersetzt, der bekommt dann logischerweise die Basisadresse $a.4 = 1.3.4$, die wieder selten zu sehen sein wird. Der nachfolgende Teil, der hinter dem ganzen if-then-else Konstrukt steht, steht dann hinter der Sprungmarke $a.3$ und hat aber intern wieder schon eine neue Basisadresse zugewiesen bekommen, weil es dann ja das vierte Statement ist, also gleichzeitg auch die Basisadresse 1.4 hat (die wiederum ist vorerst nicht sichtbar).

Analog passiert das auch bei den if-then-Statements (ohne else) und den while-Schleifen.

Abschließend will ich dir das noch kurz so versuchen zu erklären, wie man es sich vielleicht in der Klausur schnell merken kann. Ganz wichtig ist der folgende Hinweis:

Die Adressen werden immer nach dem ersten Auftreten vergeben, nicht nach dem Vorkommen der Sprungbefehle!

Das begründet auch, warum beim `JMP` nach dem then-Zweig die Adresse $a.3$ kommt und nicht $a.2$, denn $a.2$ ist schon aufgetreten in der Übersetzung des then-Zweiges. Beachte hier den Unterschied zwischen *auftreten* und *sichtbar sein* im Sinne von "es steht dann auf dem Blatt"!

Am Besten kommst du aber glaube ich, wenn du tatsächlich einfach auswendig lernst, welche Anhänge an die Basisadresse bei den einzelnen Konstrukten rankommen. Dann wird es meist auch richtig! Mach dir vielleicht eine kleine Übersicht, in der du die exakte Definition aus dem Skript ein bisschen umgangssprachlich schreibst. Und wie bereits gesagt: keine Angst, die baumstrukturierten Adressen werden sehr leger korrigiert!

Viel Erfolg bei der weiteren Vorbereitung!