No. 1

システムソフトウェア特論 試験予想問題

■並列OS

(1) マルチプロセッサのアーキテクチャをメモリアーキテクチャの観点から分類し，その各々について，OSが考慮しなければならない問題を述べよ。

・UMA型

どのプロセッサにどのプロセスを割り合てるかというスケジュールの問題がある．また，各プロセッサが共有しているデータの一貫性を壊さないようにするという問題がある．

・NUMA型（グローバルメモリあり）

上記のUMA型の問題に加え，ローカルメモリ間のデータの移動の問題．がある．

・NUMA型（グローバルメモリなし）

上記の問題に加え，グローバルメモリが存在しないため，各プロセッサの共有データをどのメモリに配置するかという問題や，OS本体をどのメモリに配置するかといった問題がある．

・NORMA型．

上記のUMA型，NUMA型の問題に加え，他のプロセッサと通さないと読めないデータの処理をどのようにするかという問題がある．

(2) カーネルデータ構造の統一性について，下記の問題に答えよ．対象マシンは，単一プロセッサシステムとする．

1) どのような場合に，統一性が壊れる可能性があるか．

・スケジューリングによるコンテキストスイッチや割り込みによる強制的なプロセッサ放棄や入出力要求を出した場合など自発的なプロセッサ放棄といった，プロセッサを放棄した場合に統一性が壊れる可能性がある．

2) また，その契機を分類し，各々の場合への対処方法を示せ．

・スケジューリングによるコンテキストスイッチ

ユーザ走行モードではコンテキストの保存で対処し，カーネル走行モードでは，コンテキストスイッチを禁止する．

No.

・ ・ ( )

・割込み

割込みハンドラへのデータはベースレベルカーネルコードがアクセスし，ベースレベルカーネルコード実行中には，割込み禁止にする．

・自発的放棄

ファイルアクセスの排他制御を行う．

(3) 共有メモリ型マルチプロセッサでのOSの実装方法を分類し，各々の方式について，実現，性能の観点から利点，欠点を述べよ．

・固定マスタスレーブ方式

単一プロセッサ用の論理が適用できるため実装が容易であるという利点があるが，性能面でMP(マスタプロセッサ)がボトルネックになるという欠点が存在する．

・浮動マスタスレーブ方式

ユーザ/カーネル間のプロセッサ間移動がないという利点があるが，モードが切り替わるとき，キャッシュが破壊されるため，キャッシュを活用できないという欠点がある．

・機能コードロック方式

OSの要素を機能分割し，並列に実行することができるが，デッドロックが発生する可能性があり，きちんとその対処をしなければならないという欠点がある．

・データロック方式

データ構造にロックをかける方式で，並列実行できる可能性は大きいという利点があるが，データ抽象型，オブジェクト指向設計であり，既存OSの再構築が必要となるため，実装が比較的困難であるという欠点がある．

■ プロセス管理とスケジューリング

(1) 共有メモリ型マルチプロセッサにおいて，(初期の) UNIX を実装したとする．本UNIXでは，1つの仮想アドレス空間と1つのコンテクストを一体化させたUNIXプロセスのみをユーザに提供しているとする．この環境において，並列処理の軽さの観点から，問題点を述べよ．また，軽い並列処理環境を提案し，その利点，欠点を述べよ．

並列処理において，並列処理させたいプロセスどうしでデータを共有させたい場合がある．しかしながらUNIXプロセスでは，1つの仮想アドレス空間しか持たないため，プロセス間通信を用いることで，データを共有することになるが，このプロセス間通信の処理は重たい．そのため，並列処理よりもデータ共有に時間を要してしまうという問題がある．ここでスレッドという，1つの仮想アドレスを他のスレッドと共有し，固有のコンテクストを持つものを考える．スレッドは仮想アドレス空間を共有しているため，スレッド間通信は高速で行うことができ，軽い並列処理が実現できるという利点がある．しかしながら，スレッドの提供方法により，実装の困難性や性能が左右されるという欠点がある．

(2) スレッドモデルを3つあげ，説明せよ．また，各々のモデルの利点，欠点を述べよ．

・カーネルレベルスレッドモデル

OSのカーネルがスレッドを提供するモデルである．仮想プロセッサがスレッドとなる．ユーザアクティビティの挙動が把握しやすいという利点があるが，以下の3つの欠点が存在する．1つ目は，アクティビティの操作がシステムコールであるため，高い性能を引き出せないという点。2つ目は，柔軟性に欠けるという点。3つ目は，より多くのスレッドモデルを包含しているため，オーバースペックになりやすいという点である．

・ユーザレベルスレッドモデル

ユーザ空間でスレッド生成や消滅などの管理を行うモデルである．ユーザ空間で実現しているため，スレッド操作を関数呼び出し程度の速度で行うことができ，速いという利点がある．しかしながら，仮想プロセッサのマッピング状況が把握できなり，仮想プロセッサのブロックが発生するといった欠点が存在する．また，仮想プロセッサの横取りによる弊害により，実行可能スレッドが実行できない，スレッドの同時実行が保証できないという問題が発生する欠点がある．

No.

・ ・ ( )

■ 同期機構

(1) キャッシュが装備されている共有メモリ型において，単にテストアンドセット命令を用いたスピンロックでは，スピードの観点から効率が悪い．なぜか？ その理由を述べよ．

プロセッサがテストアンドセット命令を実行したとき，変数lockへの書き込みが行なわれ，他のプロセッサのキャッシュから書き戻しを行った後にキャッシングを行い，さらに無効化の信号がバスを流れるというピンポン現象が起る．これによってバストラフィックが増加し，処理のスピードがおちるため，効率が悪い．

(2) 共有メモリ型マルチプロセッサにおいて，スピンロックを用いたアルゴリズムを3つあげ，説明せよ．また，各々のアルゴリズムの利点，欠点を述べよ．

・スヌーピングロック方式

ロック変数のリードのみを行い，ロック獲得の可能性があるときのみテストアンドロック命令を行う方式である．書込みがないため，コヒーレンス制御の必要がない．バストラフィックの無駄な増加は起らないという利点がある．しかし，ロック解除時に，待っていたプロセッサによる，バースト的なテストアンドロック命令のアクセスが発生してしまうという欠点を持つ．

・衝突回避ロック方式

上記のスヌーピングロック方式において，ロック解除時のバースト的アクセスを回避するため，ロック解除後のテストアンドセット命令までに遅延を挿入する方式である．スヌーピングロック方式の利点はそのままに，スヌーピングロック方式の欠点を持っていないという利点がある．

・トーナメントロック方式

上記のスヌーピングロック方式において，ロック獲得の順序をトーナメント方式であらかじめ決めておく方式である．これも，スヌーピングロック方式における，バースト的アクセスを避けることができるという利点がある．

No. ( )

(3) バリア同期において，次の問いに答えよ。

1) 別紙に示すアルゴリズムTは正しいか，正しくないか証明せよ．このとき，バリア同期は，次の2つを満たさなければならないとする。

・バリア同期の働きをする（足並みをそろえる）

・再初期化問題に対処している．

また，プロセッサ速度に関して何の制限もないとする．アルゴリズムT内の各ブロックはもろん，クリティカルセクションである．

カウンタの初期化の際に，プロセッサの横取りなどでcountが0に立ちないまま．別のプロセッサが再び，バリア同期の部分に到達してしまうと，count=Nのままなので，バリア同期を素通りしてしまい，バリア同期の働きをしない．よってこのアルゴリズムTは正しくない．

2) バリア同期を求める具体例を示せ．

配列の計算で

$B[i] = A[i+1] + A[i-1]$

$C[i] = B[i+1] + B[i-1]$

をプロセッサ1, 2, 3がそれぞれ0~29, 30~59, 60~89の要素の計算を行うとする．このとき，Cの計算を行う前にBの計算がすべて終了していないといけないので，BとCの計算の間にバリア同期を用いる．

(4) 共有メモリ型マルチプロセッサにおいて，バリア同期のアルゴリズムを3つあげ，説明せよ。また各々のアルゴリズムの利点，欠点を述べよ．

No.

( )

(4) マルチプロセッサのスケジューリングについて，考慮すべき問題点を述べよ．

マルチプロセッサであるから，プロセス間の同期に伴うオーバヘッドとコンテキストスイッチングのオーバヘッドを考慮しなければならない．また，NUMA型アーキテクチャの場合は，メモリアクセスのオーバヘッドも考慮しなければならない．単一プロセッサでは考慮していないプロセス（スレッド）の協調動作を考慮しなければならない．

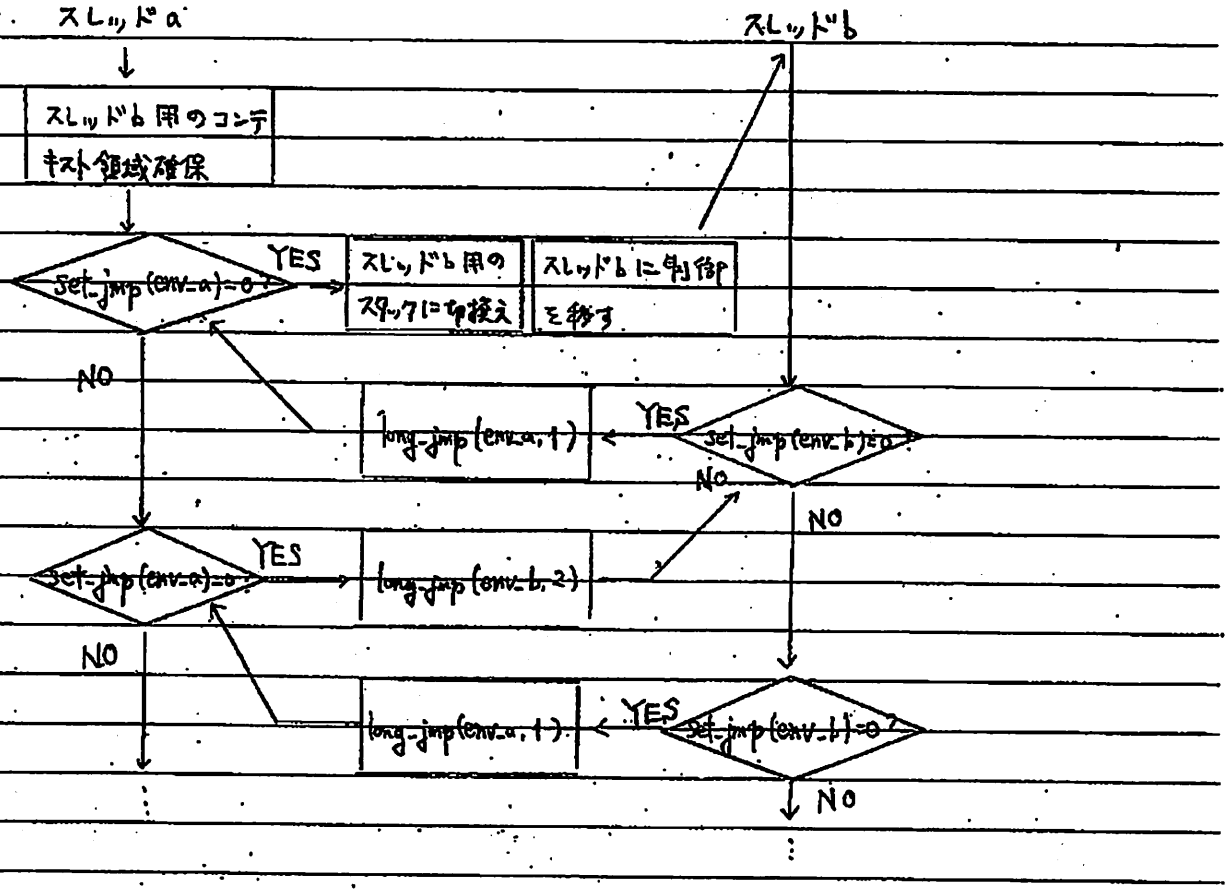
(5) コスケジューリング（Coscheduling）を説明し，実現方法を示せ．

No.

・協調型ユーザレベルスレッドモデル

ユーザレベルスレッドモデルにおいて、カーネルに仮想プロセッサのマッピング状況をユーザ空間に通知するシステムコールを追加したモデルである。カーネルレベルスレッドモデルの機能性とユーザレベルスレッドモデルの柔軟性の2つの長所をもつという利点があるが、カーネルの再構築に多大な労力がかかるという欠点がある。

(3) set_jmp 関数, long_jmp 関数を用いて、コルーチンを実現する概略フローチャートを示せ。



No.

■ リアルタイムスケジューリング

(1) 優先度固定のスケジューリングの中で，レートモノトニックスケジューリング(Rate Monotonic Scheduling)は最適なスケジューリングである．その根拠を示せ．

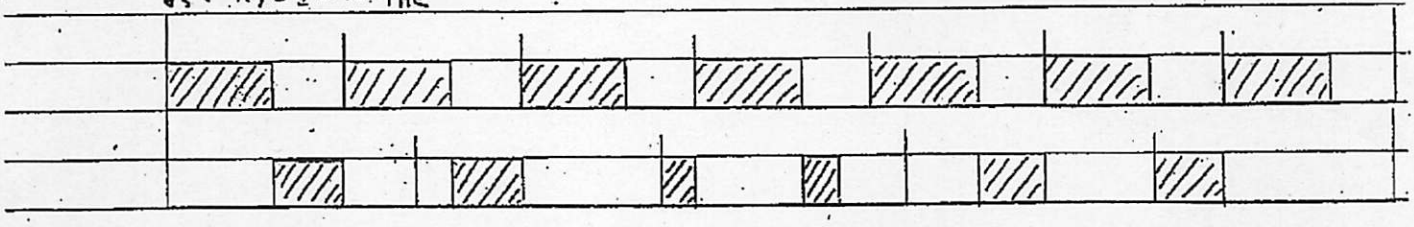
「周期の長いタスクの優先度を高くしてスケジューリングしたものがスケジューリング可能であれば，レートモノトニックスケジューリングでも必ずスケジューリング可能である」ということが証明できる．

(2) 2つの周期タスク P1，P2 のタスクセットを考える．ここで，P1 = (3,5)，P2 = (2,7) とする．但し，(実行時間，周期)．このとき，このタスクセットに関して，次のリアルタイムスケジューリングでスケジュール可能か否かを判定せよ．また，スケジュールの時間的推移を示せ．

1) レートモノトニックスケジュール

$U = \frac{3}{5} + \frac{2}{7} = \frac{15+14}{35} = \frac{29}{35} \fallingdotseq 0.8285\cdots < 0.83$

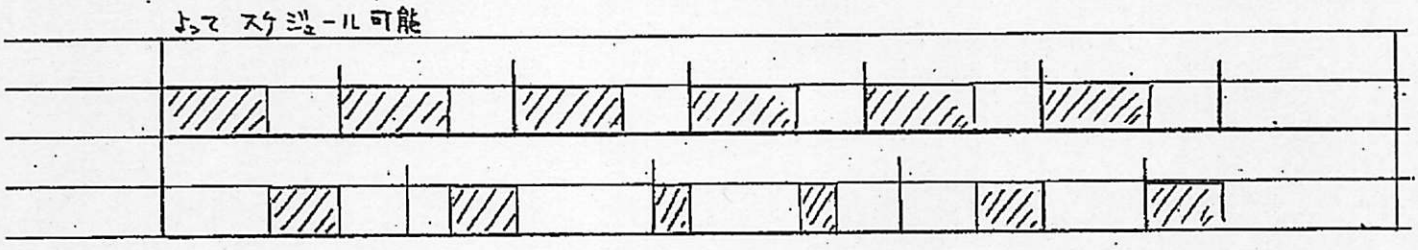
よって スケジュール可能



2) EDF スケジューリング (Earliest Deadline First Scheduling)

$U = \frac{3}{5} + \frac{2}{7} = \frac{29}{35} < 1$

よって スケジュール可能



No.

( )

■ メモリ管理

(1) キャッシュを装備している共有メモリ型マルチプロセッサにおいて，次の問いに答えよ．

1) キャッシュコヒーレンス問題とは何か？

各プロセッサのキャッシュと共有メモリ間でデータの一貫性が壊れてしまう可能性があるという問題

2) キャッシュコヒーレンス問題を解決する方法を分類して，その方法を列挙し，各々の方式の利点，欠点を述べよ．

周期の短いタスクに、高い優先度をつける

(1) レートモノトニックスケジューリング

$U = \sum_{i=1}^{m} C_i / T_i \leq m(2^{1/m} - 1)$

m: タスク数
U: プロセッサ利用率
C_i: タスクiの実行時間
T_i: タスクiの周期

タスク数mのシステムが、レートモノトニックスケジューリングできるための十分条件は

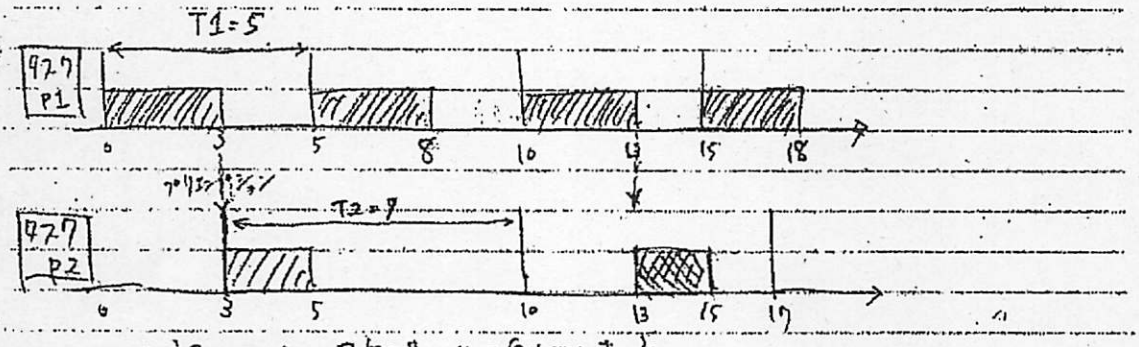
$P1 = (C_1, T_1) = (3, 5)$
$P2 = (2, 7)$

のとき、(タスク数 m=2)

$U = \frac{3}{5} + \frac{2}{7} = \frac{31}{35} = 0.885\ldots$

$m(2^{1/m} - 1) = \frac{1}{2}(\sqrt{2} - 1) = \frac{0.41\ldots}{2} = 0.205\ldots$

$U > m(2^{1/m} - 1)$ なので、スケジューリング不可能??



上記のように、スケジューリングができる

不等式は、十分条件である(必要十分条件ではない)

No. Date

(2) EDFスケジューリング

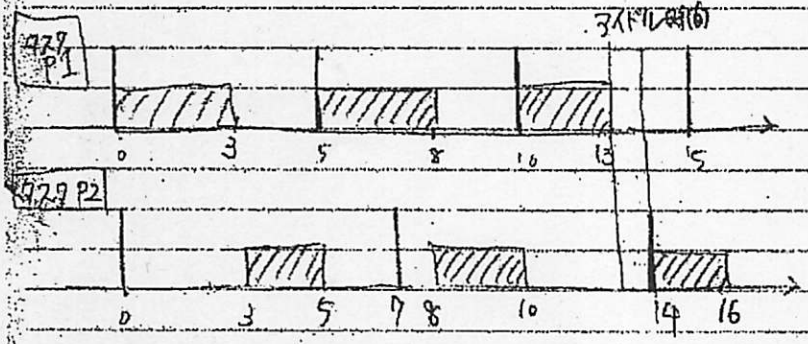
必要十分条件：任意のリリース時刻を持つタスク数 m のシステムが、EDFスケジューリングできる

$$\Leftrightarrow \sum_{i=1}^{m} \frac{C_i}{T_i} \leq 1$$

なので

$$U = \frac{3}{5} + \frac{2}{5} = 0.8857\cdots \leq 1$$

スケジューリング可能



……よく分かりませんでした。
この解答を信用しないように ね

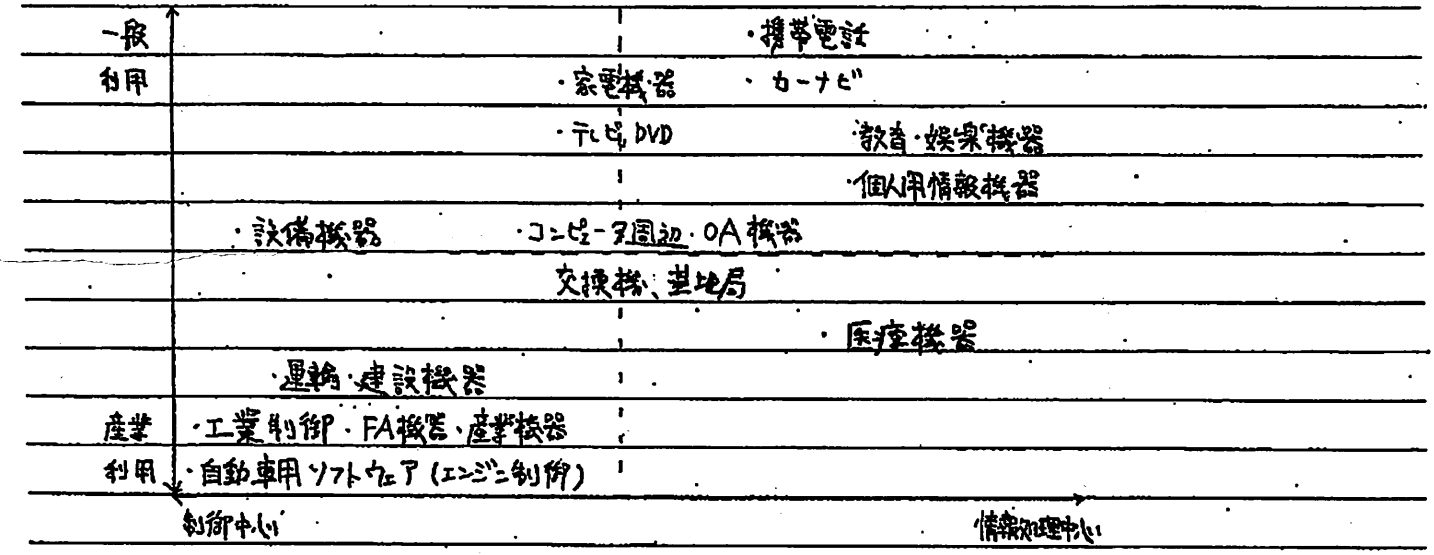
$$\frac{2+10}{35} = \frac{3}{35}$$

No.

( )

■ 組込みシステムの概論

(1) 組込みシステムを，自分なりに分類軸を考えて，分類せよ．



(2) リアルタイムシステムにおいて，タスクがデッドラインを守れなかった場合に当該システムにどのような影響があるかの観点から，リアルタイムシステムを分類し，各々の例を示せ．

・システムへの致命的な影響のないもの

携帯電話，携帯DVDビデオカメラ

・システムへ致命的な影響のあるもの

工業制御

・システムへの致命的な影響がその利用者に影響のあるもの

自動車用ソフトウェア

医療機器